

AD-A097 046

ROYAL AIRCRAFT ESTABLISHMENT FARNBOROUGH (ENGLAND)  
COMPUTER BASED TECHNIQUE FOR CONVERTING A CONTOUR MAP INTO AN E--ETC  
SEP 80 P A ROBERTS

F/6 9/2

UNCLASSIFIED

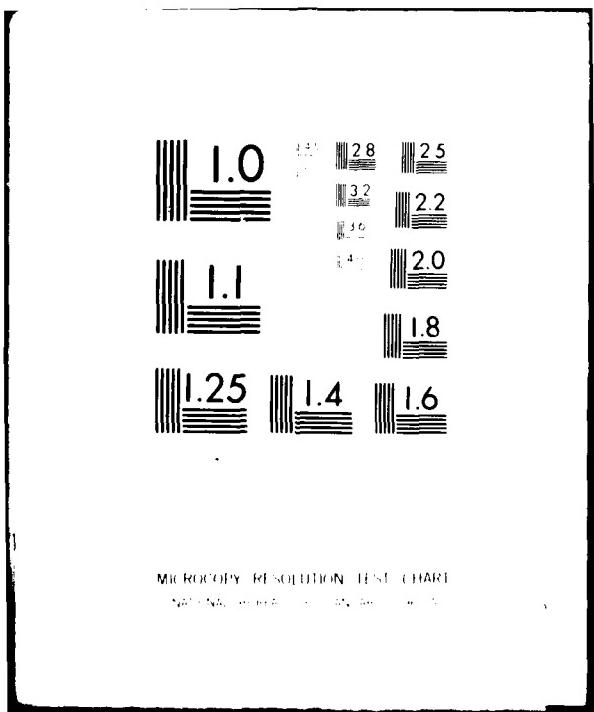
RAE-TR-80110

DRIC-DR-77598

NL

1 of 1  
28 pages

END  
DATE  
FILED  
5-81  
DTIC



TR 80110

AD A092 G40

DMC FILE COPY

UNLIMITED

BR77598✓  
TR 80110✓

1



LEVEL II

ROYAL AIRCRAFT ESTABLISHMENT

\*

DTIC  
SELECTED  
S D  
APR 13 1981

E

9 Technical Report 80110

11 September 80

COMPUTER BASED TECHNIQUE FOR  
CONVERTING A CONTOUR MAP INTO  
AN EQUISPACED GRID OF POINTS.

by

12/67

10 P.A./Roberts

14 RAE-TR-80110

18 DRIC

19 BR-77598

\*

Procurement Executive, Ministry of Defence  
Farnborough, Hants

81 3 27 160

UNLIMITED

310450

47

UDC 912 : 526.8 : 519.652 : 681.3.056 : 681.327.8

ROYAL AIRCRAFT ESTABLISHMENT

Technical Report 80110

Received for printing 17 September 1980

COMPUTER BASED TECHNIQUE FOR CONVERTING A CONTOUR MAP  
INTO AN EQUISPACED GRID OF POINTS

by

P. A. Roberts

SUMMARY

A detailed description is given of a technique for converting a contour map into an equispaced grid of points. A full description is given of program MAPGRID which converts digitized contour data into such a grid of points.

Departmental Reference: Space 586

*Copyright*  
©  
Controller HMSO London  
1980

LIST OF CONTENTS

	<u>Page</u>
1 INTRODUCTION	3
1.1 Objective	3
1.2 Outline of main program	3
2 DIGITIZATION	4
2.1 Digitization of contours	4
2.2 Grid spacing	5
2.3 Author's note	5
3 INTERPOLATION TECHNIQUES	5
3.1 The problem of interpolation	5
3.2 Interpolation requirements	6
3.3 Average tangents interpolation	7
3.3.1 The basic algorithm	7
3.3.2 Constraining the interpolation function near turning points	8
3.3.3 The problem of the single-valuedness of the interpolation function	8
4 PROGRAM DESCRIPTION	10
4.1 Program options	11
4.2 Contour data formatting	11
4.3 Program operation	12
4.4 Producing the final grid	12
4.5 Alternative version	14
4.6 The problem of interpolation outside the contour set	14
4.7 Checking the program	15
5 USING PROGRAM MAPGRID	15
6 ANCILLARY PROGRAMS	16
6.1 Program EXPAND	16
6.2 Program SECTION	17
7 CONCLUSIONS	17
Appendix A MAPGRID main program	19
Appendix B Subroutine specifications for program MAPGRID	21
Appendix C Listing of program MAPGRID	31
Appendix D Linear interpolation version of subroutine COORDS	42
Appendix E Programs to check operation of program MAPGRID	43
Appendix F Program EXPAND	44
Appendix G Program SECTION	47
Illustrations	
Report documentation page	

Accession For	
NTIS GRA&I	
DTIC TAB	
Unannounced	
Classification	
By _____	
Distribution/	
Availability Codes	
/ / and/or	
Dist	Avail
A	

Figures 1-17  
inside back cover

## 1 INTRODUCTION

### 1.1 Objective

This Report describes a technique for converting a contour map into an equispaced grid of points. The purpose of the grid representation is to enable the map data to be handled more easily in a computer and, as such, it is to form the base on which all operations on the map data are to be performed. It is assumed that the contours represent some physical quantity and thus that the variable represented by the contour height is a single-valued function of the map co-ordinates. It is also assumed that the variable is smooth and continuous.

The technique described here has applications in image processing where, for instance, it may be necessary to compare (by ratioing) an image and a contour map. A grid representation is also an essential prerequisite for degrading the spatial resolution of a contour map by convolution, thus enabling contour maps having different resolutions to be compared at a common resolution. These and other applications of this technique are to be described in a subsequent report.

Included in this Report are discussions of the problems associated with the digitization of a contour map and those of interpolation in general. The method used, the interpolation algorithms and the reasons for their use are described in detail. Also described are two operations which may be required to be performed on a grid; those of expanding (or contracting) the number of data points in the grid, and the construction of cross-sections. Listings of all programs, which are written in computer independent ANSI FORTRAN V, are given.

### 1.2 Outline of main program

Program MAPGRID transforms digitized contour data into a rectangular grid of points. The program takes two orthogonal sets of cuts across the contour map (see Fig 1); the points of intersection between these grid lines define the points at which interpolated data is required. For each grid line the points of intersection of the line with contours are calculated and then interpolation is carried out along the grid line to obtain values at the grid points lying on the line. Also produced are weighting values reflecting the confidence in the interpolated values. This procedure results in two grids of points being obtained corresponding to the two sets of cuts, together with two grids containing the associated weighting values. These weighting values are used to combine the grids to produce the final grid.

It is necessary to take two sets of cuts since it may be that some contours run parallel to one of the sets of cuts and thus would be ignored by that set unless they fell exactly on grid lines. Contours missed by one set of cuts are therefore incorporated in the orthogonal set. In this way the orientation of the cuts with respect to the contour map is not important, whilst it may be crucial if only one set of cuts were to be taken.

2 DIGITIZATION2.1 Digitization of contours

In general, the investigator will not have the contour information already digitized, but rather will only have a paper copy of the map, thus necessitating the digitization of the contours. There are several reasons why the contours should be digitized rather than digitizing along the grid lines that the computer program is going to use:

(i) It is very difficult to determine accurately the point of intersection of a grid line with a contour if the two cross at a very oblique angle.

(ii) Digitization around a contour is often much easier if a manual digitizing table is to be used, since the exact position of digitized points along a contour is not important, whereas the exact point of intersection between a grid line and a contour is much more important.

(iii) If in the future it is required to generate grids at other larger spacings, then it is not necessary to redigitize the map.

(iv) It is easier to check for any errors in the digitized data if the digitization is performed around contours, because the contours can easily be plotted and compared with the original. With the alternative method a complex contour threading operation would have to be performed before the contours could be plotted, which would in itself give rise to some differences with the original contours.

It is necessary for the investigator to decide the interval at which points on a contour should be digitized. The evaluation of this interval involves an estimate being made of the interval along a contour that can be represented by a straight line. This interval is dependent upon the grid spacing and the curvature of the contour in question; the smaller the grid spacing or the greater the curvature of the contour, the smaller this interval must be in order to represent the contour to the required accuracy. More precisely, if  $d$  is the digitization interval,  $r$  is the radius of curvature of the contour (which generally is a function of position along the contour) and  $s$  is the maximum permissible deviation of a straight line, joining two points on the contour, from the contour, then  $d \approx \sqrt{8rs}$ , where  $d$  is such that the angle the contour turns through is small. The author has found that for his work it is adequate that  $s$  be half the grid spacing, but this may not be acceptable for all applications.

If the digitization interval is greater than  $d$  then the contour is under-digitized, and if it is less it is over-digitized. The only consequence of the latter is that the number of data points describing the contour is greater than necessary. The contour fitting algorithm (described in section 3.3.3) is considerably better than linear interpolation and is adopted to make a 'best' estimate, which is necessary with under-digitized data.

The digitization of contours can take either of the following forms (amongst others):

(i) Constant separation between data points.

(ii) Variable density system with the interval depending upon the curvature of the contour. This reduces the number of points required to describe a contour.

If it is decided to opt for a constant separation between data points then the map will be correctly digitized if the evaluation of the digitization interval is based upon the smallest radius of curvature of any part of any contour. The variable density method requires a continual evaluation of the curvature of the contour and is best suited to automated digitization. If the digitization is being performed manually using this method then the investigator must make a subjective estimate of the required digitization interval.

## 2.2 Grid spacing

The main information content of a contour map is contained in the relative positions of the different contours and the large scale features shown by them, rather than in any small scale detail that may be shown by individual contours. Thus to retain the main information content when transforming a contour map into a grid, the grid spacing must be half the value of the minimum separation between any two contours or of large scale loops of the same contour. Any detail on a scale smaller than this will not be contained in the grid. More generally, the grid spacing should be at least half the size of the smallest feature of interest.

## 2.3 Author's note

The above discussion is intended to illustrate some of the basic problems involved in digitizing contours. It is not intended to be an exhaustive or highly rigorous treatment of the subject.

# 3 INTERPOLATION TECHNIQUES

## 3.1 The problem of interpolation

It is often desired to estimate the value of a function, which is sampled at certain discrete points, at some intermediate points. This process is known as interpolation. Although any set of data points can be interpolated by an infinite number of different functions, one normally requires that interpolated values are reasonable. What is believed to be reasonable depends upon the laws and processes underlying the data values. It must be realised that unless a precise mathematical function expressing those laws can be formulated, then no method of interpolation can accurately reproduce the missing data points. In many instances it may be that whilst the nature of the processes are understood they are not expressed in mathematical form, *e.g.* surface topography considered as an outcome of geological, climatic and cultural factors. In such instances it is not possible to distinguish analytically and quantitatively between the desirability of different algorithms. Instead, the choice has to rely on the judgement of the investigator himself, *i.e.* some rationalisation of a comparison between an achieved interpolation value and a subjective impression of what the value would have been at that point. In cases where the nature of the underlying processes and laws are not known the normal criteria of smoothness and simplicity have to be adopted. These are conditions which are

intuitively desirable in the absence of detailed information as to the nature of the fit required.

Interpolation functions can be of two types: local or global. A local function uses only data points near to the point at which an interpolated value is required, eg linear interpolation. A global function, however, utilizes all of the data points in the data set, eg cubic spline interpolation. With this function distant data points make a contribution to the interpolated value. Also, if the data set contains a large number of points then this type of function generally involves a large amount of computation. In cases where there is a mathematical function expressing the variable it may be that a global function is the most appropriate, but, in instances where the only requirements are for smoothness and simplicity, then a local function can be adopted. Indeed, the adoption of such a requirement is desirable if there is no knowledge of the underlying processes; the adoption of a global function, whereby distant data points can affect the interpolated values, unnecessarily implies more correlation between data points than is warranted. Obviously if it is known that there is no correlation between near and distant data points then a local function must be adopted.

The suitability of a particular function is also dependent upon the density of the data points. For very closely spaced data the nature of the interpolation function is largely unimportant and it may be that linear interpolation is entirely appropriate. In general, the more widely spaced the data the fewer functions will prove acceptable. A further (practical) consideration is the amount of computation involved with different functions if large numbers of data points and interpolated values are required. Thus a compromise may have to be made between the suitability of a particular algorithm and maintaining the amount of computation at a manageable level.

In choosing an interpolation function it should be realised that in general no method of interpolation is precisely accurate except at the points through which the function has been fitted. If little is known about the exact properties of the variable involved, then the assessment of the suitability of any interpolation function is largely left to the investigator's intuition. Thus no general rule as to the applicability of a particular function can be given, so each problem must be considered individually.

### 3.2 Interpolation requirements

There are two different interpolation procedures that have to be carried out:

- (i) Interpolation along a contour.
- (ii) Interpolation along a grid line using contour cut data.

As mentioned earlier, the choice of an interpolation function is dependent upon the density of the data. For interpolation around a contour, linear interpolation will be appropriate if the spacing between the data points is sufficiently small; however, if this is not the case, a more appropriate function is required. This function should possess the following properties:

- (i) Continuous.
- (ii) Continuous first derivative.
- (iii) Local.
- (iv) Independent of the co-ordinate system.

The first two requirements are those normally adopted where no knowledge of the underlying laws and processes is assumed and they are appropriate here, because contours are expected to be smooth and continuous. Since it is to be taken there is no *a priori* knowledge of the properties of the variable involved, it is required that the interpolation algorithm be local rather than global, *i.e.* dependent only upon the data points near to the points at which interpolated values are required. This also means that the interpolation function is going to be reasonably efficient for implementation on a computer. The fourth requirement is introduced because, for contours, the co-ordinate system (generally) does not have any significance for the variable involved, thus the shape of an interpolated curve should be independent of the orientation of the axes of the map co-ordinate system. The choice of a suitable function for contour data is further complicated in that, in cartesian co-ordinates, infinite gradients ( $dy/dx = \infty$ ) can be encountered, and the function may have to be multi-valued under certain circumstances.

When interpolating intermediate values along grid lines, infinite gradients and multi-valuedness will not be encountered and there is no need for the function to be independent of the orientation of the co-ordinate system because the choice of axis for the contour 'height' is not arbitrary; no other choice of axis could reasonably be adopted for physically realisable data. Thus for interpolation along a grid line a function is required that is continuous, with a continuous first derivative and is local. In addition, the function must be constrained at turning points (*i.e.* maxima and minima), so that interpolated values do not reach a neighbouring contour level.

With these considerations in mind, it was decided to adopt an average tangents form of interpolation.

### 3.3 Average tangents interpolation

#### 3.3.1 The basic algorithm

The method, which is applicable to any data set for which  $y$  is a single-valued function of  $x$ , is illustrated in Fig 2.  $T_i$  is the connection vector between adjacent data points and the tangent of the angle between the positive  $x$  axis and  $T_i$  is given by:

$$\tan(\theta_i) = \frac{y_{i+1} - y_i}{x_{i+1} - x_i} \quad i = 1, 2, \dots, n-1$$

where  $n$  is the number of data points in the data set.

An average is now taken of pairs of consecutive tangents as follows:

$$A_i = \frac{\tan(\theta_{i-1}) + \tan(\theta_i)}{2} \quad i = 2, 3, \dots, n-1$$

For the first and last intervals:

$$A_1 = \tan(\theta_1) \quad A_n = \tan(\theta_{n-1}) .$$

The average tangent  $A_i$  represents the mean gradient at the  $i$ th data point. Since an interpolation function is required that is continuous with a continuous first derivative the function can be represented by a third order polynomial between points  $i$  and  $i+1$ , i.e.

$$y = a(x - x_i)^3 + b(x - x_i)^2 + c(x - x_i) + d .$$

The coefficients of the polynomial can be found by choosing the first derivative of the polynomial at the points  $i$  and  $i+1$  as being the values of the average tangent at these points, whence:

$$a = \frac{A_i(x_{i+1} - x_i) + A_{i+1}(x_{i+1} - x_i) - 2(y_{i+1} - y_i)}{(x_{i+1} - x_i)^3}$$

$$b = \frac{3(y_{i+1} - y_i) - 2A_i(x_{i+1} - x_i) - A_{i+1}(x_{i+1} - x_i)}{(x_{i+1} - x_i)^2}$$

$$c = A_i$$

$$d = y_i .$$

With this method a different third order polynomial is fitted to each interval.

### 3.3.2 Constraining the interpolation function near turning points

The interpolation function for contour cut data has to be modified so that interpolated values remain within the interval defined by the two points between which the function is valid, or, if this interval is zero, do not reach a particular value.

Fig 3a&b (bold lines) illustrates cases where such modification is required. The  $x$  and  $y$  values at the turning points (denoted by subscript  $t$ ) are obtained from the equations:

$$0 = 3ax_t^2 + 2bx_t + c$$

and  $y_t = ax_t^3 + bx_t^2 + cx_t + d .$

If both values of  $x_t$  lie between  $x_i$  and  $x_{i+1}$  then the situation is as illustrated in Fig 3a, and if only one value of  $x_t$  satisfies this criterion then the situation is as depicted in Fig 3b. The former case will be discussed first.

If both values of  $y_t$  lie between  $y_i$  and  $y_{i+1}$  the interpolation function is not modified; otherwise the range is divided into three, defined by the points given below, and new interpolation functions fitted.

$$\begin{aligned}
 x &= x_i & A &= A_i & y &= y_i \\
 x &= x_{t1} & A &= 0 & y &= y_{t1} & \text{if } y_{t1} < y_{i+1} \\
 &&&& y &= y_i + \alpha(y_{i+1} - y_i) & \text{if } y_{t1} > y_{i+1} \\
 x &= x_{t2} & A &= 0 & y &= y_{t2} & \text{if } y_{t2} > y_i \\
 &&&& y &= y_{i+1} - \alpha(y_{i+1} - y_i) & \text{if } y_{t2} < y_i \\
 x &= x_{i+1} & A &= A_{i+1} & y &= y_{i+1}
 \end{aligned}$$

( $\alpha$  is a positive constant less than 1 such that  $y < y_{i+1}$  for the precision with which interpolated values are required. Generally  $\alpha = 0.99$  is a suitable choice.)

If the new interpolation function does not remain between  $y_i$  and  $y_{i+1}$  (as illustrated in Fig 3a by the dashed line) then the interpolated  $y$  values near the turning point that lie beyond either  $y_i + \alpha(y_{i+1} - y_i)$  or  $y_{i+1} - \alpha(y_{i+1} - y_i)$ , whichever is appropriate, are set to that value. This situation will arise if  $A_i + 3\alpha\left(\frac{y_{i+1} - y_i}{x_{i+1} - x_i}\right)$ , a condition that for most types of data should rarely occur. The procedure just described gives rise to discontinuities in the gradient, however in practice since interpolated values are required at discrete points this somewhat alleviates the problem.

If there is only one value of  $x_t$  in the range  $x_i$  to  $x_{i+1}$  it may be required that interpolated values are not to exceed a particular value of  $y$ , say  $y_o$  (ie the next contour level). If the interpolation function does not reach  $y_o$  then the function remains unchanged, otherwise two new functions are fitted using the following points:

$$\begin{aligned}
 x &= x_i & A &= A_i & y &= y_i \\
 x &= x_t & A &= 0 & y &= y_i + \alpha(y_o - y_i) \\
 x &= x_{i+1} & A &= A_{i+1} & y &= y_{i+1}
 \end{aligned}$$

If the new function does not remain between  $y_i$  and  $y_i + \alpha(y_o - y_i)$  (as illustrated in Fig 3b by the dashed line), then interpolated values lying beyond  $y_i + \alpha(y_o - y_i)$  are set to that value. It can be shown that for contour cut data, where  $y_i = y_{i+1}$ , the interpolation function must take the form shown in Fig 3b (ie remain above or below  $y_i$  between  $x_i$  and  $x_{i+1}$ ) since  $A_i$  and  $A_{i+1}$  have opposite senses (if one is positive the other is negative; zero can be regarded as positive or negative depending upon what is necessary for  $A_i$  and  $A_{i+1}$  to be of opposite sense).

### 3.3.3 The problem of the single-valuedness of the interpolation function

The average tangents algorithm is not independent of the orientation of the co-ordinate system and in some cases the algorithm fails completely, e.g. when infinite gradients are involved, when  $y$  is a multi-valued function of  $x$ . This problem can however be overcome. Rather than considering  $y$  changing with respect to  $x$ ,  $x$  and  $y$  are considered independently as changing with respect to the separation ( $s$ ) between adjacent data points.

Define

$$s_i = \left( (x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2 \right)^{\frac{1}{2}} \quad i = 1, 2, \dots, n-1$$

$$m_i(x) = \frac{x_{i+1} - x_i}{s_i} \quad \text{and} \quad m_i(y) = \frac{y_{i+1} - y_i}{s_i} .$$

The average tangents algorithm can now be applied to  $x$  and  $y$  separately as a function of  $s_i$  with parameters  $m_i$  being substituted for  $\tan(\theta_i)$ . Two equations are thus obtained:

$$x = a_x s^3 + b_x s^2 + c_x s + d_x$$

and

$$y = a_y s^3 + b_y s^2 + c_y s + d_y$$

$$\text{where } s = \left( (x - x_i)^2 + (y - y_i)^2 \right)^{\frac{1}{2}} .$$

Thus to derive the  $y$  value corresponding to a particular  $x$  value, the value of  $s$  corresponding to the  $x$  value is found by solving the first equation (with the condition that  $0 < s < s_i$ ), and then substituting the value for  $s$  in the second equation to obtain  $y$ .

With this procedure the algorithm never fails because 'infinite' gradients are never encountered since  $s_i > 0$  ( $s_i = 0$  corresponds to two adjacent data points being coincident). The algorithm is independent of the orientation of the co-ordinate system since  $x$  and  $y$  are treated in exactly the same manner as a function of the co-ordinate free quantity  $s$ .

#### 4 PROGRAM DESCRIPTION

Details of the main program such as dimensions of arrays, initialisation of parameters, and program units called are given in Appendix A. Detailed specifications of all subroutines are given in Appendix B, and a complete program listing is given in Appendix C. In the program listing the parameters initialised at the start of the main program are those for the example discussed in this Report.

#### 4.1 Program options

Limits to the allowed values of interpolated map data have to be set. If no limits are to be placed on interpolated values then the parameters setting the limits should be set so that they are well away from any of the contour values and anticipated interpolated values. The parameter FRACTN sets how closely interpolated values can approach the next contour level at turning points. It is recommended that this be set at 0.99, as discussed in section 3.3.2, so that interpolated values just fall short of the next level.

Maps may be classified into two categories for the problem being considered here, closed or open. A closed map is defined to be one where all contours start and finish within the region being considered (as in Fig 1) and an open map is one where only some contours start and finish within the region (as in the area enclosed by the dashed rectangle in Fig 1). The operation of the program is slightly different for the two types of map, this being related to combining the values obtained from the two sets of grid lines and with the problem of interpolation outside a dataset. These are discussed in more detail in sections 4.4 and 4.6 respectively. The two modes are selected as follows:

(i) Closed maps. IROUTE = 1. ZE = value to which all points beyond the outermost contour are to be set.

(ii) Open maps. IROUTE ≠ 1. ZE = value to which all points through which pass x and y direction grid lines that do not intersect any contours.

#### 4.2 Contour data formatting

The program requires that the contour data be in a specific form. Each contour should have a header which should then be followed by the x and y co-ordinates of each point, there being one point per line. The header contains three parameters: the contour value, the number of points in the contour and the type of contour. The first two parameters are self-explanatory but the third requires further comment.

Contours can be of two types, either closed such as a circle or open such as a line. It is necessary to know which of these types a contour is because of the problems associated with interpolation in the end intervals of a dataset. The average tangents algorithm (section 3.3.1) sets the average tangent at the end data points to be the gradient between the two points forming the end interval, but for the case of a closed contour a better estimate is possible because the data points beyond the end interval of the dataset are known. Thus for closed contours (for which it is assumed that the first and last points have the same co-ordinates) the first two points are also added to the end of the dataset and then interpolation is only considered between the second and second to last points on the contour. This procedure cannot of course be carried out with open contours, yet it would be convenient if both types could be treated in the same manner. Thus for open contours an extra data point is added at each end using linear interpolation so that the average tangents at what are now the second and second to last points remain the same. Therefore, after a contour has been read in, it is reformatted in one of the above ways by subroutine IDENT.

#### 4.3 Program operation

The program starts by initialising all the constants and calculated the  $x$  and  $y$  values of the grid lines. The values of the points in the grid produced by taking cuts in the  $y$  direction are then calculated.

Contours are read in and processed one at a time. The contour data is first checked by subroutine CHECK to ensure that no two adjacent points have the same co-ordinates, which would cause the contour interpolation algorithm to fail. The contour data is then reformatted, as described above, by subroutine IDENT. Following this, subroutine COORDS is called to calculate the points at which the  $y$  direction grid lines intersect the contour; the algorithm described in section 3.3.3 is used here. This procedure is repeated for each contour, after which subroutine HEIGHT is called to sort the different contour values, together with the upper and lower limits to the map data, into ascending order. Each of the  $y$  direction grid lines is now taken in turn. Subroutine SELECT is called to obtain all the points of intersection of contours with a chosen grid line and these points are then arranged into ascending order in the  $y$  direction by subroutine ORDER. Subroutine CHECK2 is then called to ensure that no two adjacent data points have the same  $y$  value, which would cause the subsequent interpolation algorithm to fail. Subroutine AVTAN calculates the values on the grid along the grid line using the average tangents algorithm described in sections 3.3.1 and 3.3.2. The associated weighting factors for each of the grid points are obtained by a call to subroutine WEIGHT.

The above procedure is repeated for the grid lines in the  $x$  direction, the only differences being that subroutine HEIGHT is not called, and as each new line is generated it is combined with the first grid to produce the final grid. On completion of this the final grid is then written to a file with the data being written from the top of the grid downwards, ie as a series of  $x$  direction grid lines in descending values of  $y$ .

#### 4.4 Producing the final grid

Associated with each interpolated value is a weighting factor. The weighting factor reflects the fact that an interpolated value is most likely to be reliable where the original data is most closely spaced. The weighting factor is given by:

$$w_x = \frac{1}{|x_{i+1} - x_i|}$$

where the point at which an interpolated value is required lies between  $x_i$  and  $x_{i+1}$ . If the point lies outside the range of the dataset then  $w_x = 0$ .

Three methods of combining the two grids were investigated:

$$(i) G_f = \frac{(G_x + G_y)}{2}$$

$$\begin{aligned}
 \text{(ii)} \quad G_f &= \frac{(G_x W_x + G_y W_y)}{(W_x + W_y)} & (W_x + W_y > 0) \\
 G_f &= \frac{(G_x + G_y)}{2} & (W_x + W_y = 0) \\
 \text{(iii)} \quad G_f &= G_x & (W_x > W_y) \\
 G_f &= G_y & (W_x < W_y) \\
 G_f &= \frac{(G_x + G_y)}{2} & (W_x = W_y)
 \end{aligned}$$

where  $G_x$  and  $G_y$  are the values at a particular grid location produced by taking cuts parallel to the  $x$  and  $y$  axes respectively,  $W_x$  and  $W_y$  are the corresponding weighting factors, and  $G_f$  is the final value.

The desirability of each of these methods was judged on the difference between the grid obtained from a series of contours of a two dimensional Gaussian and a grid derived analytically. The contours were produced at intervals of 25 between 25 and 225 and both grids were set to 0 beyond the outermost contour. Figs 4 and 5 show the two grids produced by the two sets of cuts and Figs 6, 7 and 8 show the differences between the program grid and the analytic grid for the three methods; differences greater than  $|9|$  are set to 9.

Fig 6 lends weight to the reasoning for the weighting factors used in methods 2 and 3, since the differences are a minimum  $45^\circ$  away from the  $X$  and  $Y$  axes, where the spacing of the original data is the same for each set of grid lines. Fig 7 shows a considerable improvement in the agreement but suggests that considerably greater weighting should be given to the larger of the two weighting values. This is taken to the extreme in the third method, producing a further improvement.

It is the third method, with some modification, that has been adopted to combine the values produced by the two sets of grid lines. The weighting values are calculated as before, except that if  $A_i = A_{i+1} = 0$  and  $z_i = z_{i+1}$  then  $W = -1$ , and if the grid line under consideration does not intersect any contour then  $W = -2$ . The method for combining the values is dependent upon the type of map, and is as carried out according to the following sets of rules:

<u>Closed maps</u>		<u>Open maps</u>
$G_f = \frac{(G_x + G_y)}{2}$	$(W_x = W_y)$	$G_f = \left( \frac{G_x + G_y}{2} \right)$
$G_f = G_x$	$(W_x = -2)$	
$G_f = G_y$	$(W_y = -2)$	$G_f = G_x (W_x = -1, W_y = -2 \text{ or } 0)$
$G_f = G_y$	$(W_x = 0)$	$G_f = G_y (W_y = -1, W_x = -2 \text{ or } 0)$
$G_f = G_x$	$(W_y = 0)$	$G_f = G_x (W_x > W_y)$
$G_f = G_x$	$(W_x > W_y)$	$G_f = G_y (W_y > W_x)$
$G_f = G_y$	$(W_y > W_x)$	

The extra conditions arise from the following considerations. In Fig 9a the grid line yy intersects the contour with height 2 at points a through e. Interpolated values along yy between b and d are given as 2 but, by inspection, the variable should be less than 2 between b and c and greater than 2 between c and d. The correct sense for the interpolated data is obtained by using interpolated data produced by the orthogonal set of grid lines. If the weighting of a point is zero then it lies outside the dataset defined by the intersection of the associated grid line with contours, ie for a closed map the point lies beyond the outermost contours. The value of the final grid at that point should therefore have the value ZE which is to be assigned to such regions. Applying the same procedure to open maps would produce undesirable effects, for example all values within the shaded area of Fig 9b would be the same.

The final grid produced by combining those shown in Figs 4 and 5 with this method is shown in Fig 10.

#### 4.5 Alternative version

The interpolation algorithm described in this Report for carrying out interpolation with contour data is adopted so as to give a 'best' estimate. However, if the contours have been digitized such that linear interpolation is appropriate, an alternative version of subroutine COORDS employing linear interpolation is available. This subroutine is listed in Appendix D and can be simply substituted for that given in Appendix B, omitting subroutines INTERP and CUBIC which are no longer required.

#### 4.6 The problem of interpolation outside the contour set

In the case of closed maps, the user of the program is required to decide what values should be assigned to grid points lying outside the outermost contour. Two possible options are:

- (i) Setting all values to the same value, say that of the base level or the outermost contour.
- (ii) Extrapolating smoothly to some defined level.

The first option is easy to implement and is available with this program, but the second is much more difficult. It should be realised that one cannot reliably extrapolate beyond the outermost contour unless the nature of the variable in the region is known. It may be that the user knows that the variable tends to some base level and so wants to extrapolate smoothly to this level for 'aesthetic' reasons. In this situation one very satisfactory method is for the user to define a 'false' contour beyond the outermost contour, where it is estimated that the variable reaches the base level, define a second false contour (the shape of which is not important) outside the first and then run the program with all values beyond the false contours being set to the base level. The second false contour ensures that along a grid line the interpolation curve will have a gradient of zero when it reaches the base level and that areas of constant value do not occur in the region between the first false contour and the outermost contour of the original data. (This latter point is discussed in more detail in section 5.) The difficulty of trying to extrapolate analytically is that the contours defined by the extrapolation routines may be far from smooth, which is undesirable if the original map contours are smooth.

If the first of the above options is applied to open maps, then the areas enclosed by a corner of the map and the contour closest to that corner, will be areas of constant height. Thus, if possible the user should enlarge the area over which contours are digitized, such that the required area is contained within the final grid but does not encompass any of the areas of constant height occurring at the corners.

#### 4.7 Checking the program

Two programs are listed in Appendix E to enable a user to test program MAPGRID. The program DRIVER1 will produce the contour data that was used in running MAPGRID to produce the grid shown in Fig 10. Program DRIVER2 will produce the same grid analytically, the difference between the two being that shown in Fig 8.

#### 5 USING PROGRAM MAPGRID

For most contour maps that it is anticipated that a user will encounter the method described in this paper will work well. There are, though, several situations where the resultant grid is not particularly reliable. Here some examples of this will be discussed together with some methods for overcoming the difficulties.

In Fig 8 the greatest difference between the interpolated and analytic grid is 5, compared with a contour interval of 25, which generally would be considered to be good agreement since no *a priori* knowledge of the variable is assumed, other than it be smooth and continuous. However, it is worth examining a diagonal cross-section which passes through the centre of the grid. The resultant curve (which is produced by program SECTION described in section 6.2) is shown in Fig 11, where it can be seen that there is a slight 'peculiarity' in the curve near contour level 200. The cross-section is at 45° to the two sets of grid lines and it is evident that if the cross-section had been taken parallel to a set of grid lines the feature would not occur. The reason for the occurrence of this feature can best be illustrated by reference to Fig 12. For grid line aa the interpolated values in the range bb become increasingly unreliable as bb , and hence d , increases. Interpolated values are most unreliable for the largest values of d which satisfy the condition that d be less than  $r_2 - r_1$  , but interpolated values in the interval bb are not incorporated in the grid if the weighting of the points associated with the orthogonal set of grid lines is greater. The weighting values associated with the final grid shown in Fig 10 can be seen in Fig 13. The weighting values indicated by 1 lie near point X in Fig 12. By inspection it can be seen that for certain values of  $r_2$  and  $r_1$  , X does not lie inside  $r_2$  , the criterion for this being that  $r_2/r_1 < \sqrt{2}$  . This is why no other feature similar to that which occurs at contour level 200 exists on the cross-section, ie

$$\frac{r_{175}}{r_{200}} = 1.316 \quad \text{whereas} \quad \frac{r_{200}}{r_{225}} = 1.681$$

where the subscripts indicate the contour heights. Thus, when taking a series of cuts across a map it is preferable to take these parallel to a set of grid lines.

Another difficulty which may be encountered is best illustrated by reference to Fig 14a. In this figure there are two sets of closed contours indicated by A and B, set A consists of three contours and set B of only one contour. By inspection it can be seen that within the contour of set B the variable is greater than 1 and less than the next contour level of 2, and outside sets A and B the variable is less than 1 but greater than or equal to the base level of 0. It is not possible however to make any reliable estimate as to the magnitude of the departure of the variable from 1 in these cases. Fig 14b&c indicates the reliability of interpolated values obtained by taking cuts across the map in the x and y directions. Fig 14d shows the values obtained after the two grids have been combined, from which it can be seen that reliable information as to the shape of the lowest contour is lost. The contour with value 2 is correctly described and all interpolated values within it are considered reliable.

The lowest contour level and the sense, although not the magnitude, of the interpolated data can be correctly described using the following method. The contour map is considered as consisting of two separate contour maps, these being the contour sets A and B. Another contour with a value between 0 and 1 is drawn outside the two sets as shown in Fig 15. If program MAPGRID is now run (IROUTE set to 1, ZE set to the base level of 0), the resultant grid will be reliable for grid values greater than or equal to 1. It is suggested when drawing the false contours that the closer the chosen contour level of the false contour is to the value of the outermost contour the closer the false contour should be drawn to that contour. In drawing a false contour close to the outermost contour a user is most likely to estimate accurately the behaviour of the variable beyond the outermost contour. This procedure of drawing false contours is recommended even if the contour map shown in Fig 14a were to consist only of contour set A, since from inspection of Fig 14d certain parts of the interval between contour levels 1 and 2 contain areas of constant height not implied by the data.

The above discussion is by no means exhaustive, but is merely included to indicate some problems which can be encountered and how it may be possible to overcome them. Clearly the user should study any map before using the technique described in this Report to ascertain whether it is desirable to introduce any false contours or to consider the map in several parts.

## 6 ANCILLARY PROGRAMS

Two further programs have been written to carry out operations on the grid produced by program MAPGRID. One of the programs expands the number of data points in the grid whilst the other enables a cross-section to be taken through it.

### 6.1 Program EXPAND

One of the requirements that the author has is to display the grid on a TV monitor so as to visually inspect the data. The number of pixels required to fill the screen has been  $512 \times 512$ , a number considerably greater than that needed in other

computer processing of the grid. This program has therefore been written to expand (or contract) the number of data points in either or both the x and y directions. The different expansion factors for the two directions means that the aspect ratio (width/height) of the grid can be altered so that the resulting image appears correctly proportioned on a TV monitor (normal aspect ratio of 1.33).

Care has to be exercised when interpolating data that has already been interpolated. In general it is accepted that in such instances it is best to use the same interpolation algorithm as was used to produce the interpolated data. Thus in this program the average tangents algorithm of section 3.3.1 was used, but it omits the 'logic' which prevents data at turning points reaching the next contour level as information concerning the next contour level is not contained in the grid.

The operation of the program is straightforward, the user only having to specify the expansion factor required in each of the directions. The number of points down each column of the grid is first adjusted to the required number and then the procedure is repeated for each line, each line being written to an output file as it is computed. A description and a listing of the program is given in Appendix F.

#### 6.2 Program SECTION

A common requirement is to look at a section across a grid. This program enables any cross-section in any direction to be taken. Again, for consistency, the average tangents algorithm is used for interpolation.

The method adopted is as follows. The co-ordinates at which interpolated values along the cut are required are calculated from the co-ordinates of the two ends of the cut and the number of points required in the cut. If the number requested is less than two then it is assumed that points are required at the same interval as the grid spacing. Values at the required points (\*) in Fig 16 are evaluated as follows. Four points (o) are calculated with the average tangents algorithm using the four points linked by the dashed line. These four points, linked by the dotted line, are then used to obtain a value at the required point (\*). This procedure is then repeated in the orthogonal direction and a second estimate obtained. The final value is taken as the average of the two estimates. In practice it is found that the difference between the two estimates is insignificant if the grid has been produced by program MAPGRID. With this method, interpolated values cannot be obtained in the end intervals, so values required in these intervals are initially ignored and only those for which the above method is applicable are derived. The values required in the end intervals are obtained from linear extrapolation of the data previously derived.

A description and a listing of the program is given in Appendix G.

#### 7 CONCLUSIONS

This Report has described a technique for converting digitized contour data into an equispaced grid of points, and a computer program MAPGRID which performs this transformation has been included. It should be realised that the computer programs described in this Report are not designed either for maximum efficiency or minimum storage requirements. They are included so that the reader can use them to assess the technique and are written for clarity so that they can easily be modified if required. Some applications of this technique are to be described in a subsequent report.

Appendix AMAPGRID MAIN PROGRAMA.1 Function

The main program initialises constants, supervises the flow, controls all input and output and combines the data from the two sets of cuts to produce the final grid.

A.2 Dimensions of arrays

F	}	maximum number of points in any contour
G	}	
X	}	maximum number of points of intersection of a set of grid lines with the contours
Y	}	
Z	}	
A	}	maximum number of contour crossings of any grid line
S	}	
W	}	
U		number of grid lines parallel to the y axis (IDIM)
V		number of grid lines parallel to the x axis (JDIM)
WT		the larger of IDIM and JDIM
H		number of contours +2
AA	}	IDIM, JDIM.
WA	}	

A.3 Constants to be initialised

MAX	maximum number of contour crossings of any grid line (= dimension of A, S and W)
KMAX	maximum number of points of intersection of a set of grid lines with the contours (= dimension of X, Y and Z)
IDIM	number of grid lines parallel to the y axis
JDIM	number of grid lines parallel to the x axis
IROUTE	control parameter: = 1 for closed maps, #1 for open maps
ZE	value to be assigned to points outside the contour map
WW	grid spacing
XST	x co-ordinate of the origin of the grid
YST	y co-ordinate of the origin of the grid
HMIN	minimum allowed value of map data
HMAX	maximum allowed value of map data
FRACTN	normally set at 0.99, see section 3.3.2

#### A.4 Program units

CHECK ensures that no two adjacent data points are coincident  
IDENT reformats the contour data  
COORDS evaluates the co-ordinates at which y direction grid lines intersect a contour  
NFIX gives the integer value of a number, the value being rounded down to the lower integer  
INTERP calculates the co-ordinates at which a line parallel to the y axis intersects a contour between two points  
CUBIC finds the roots of a third order polynomial in x in a specified range  
HEIGHT sorts the contour values into ascending order  
SELECT selects data points with a given x co-ordinate  
CHECK2 ensures that no two data points on a grid line are coincident  
ORDER sorts values in an array into ascending order  
AVTAN calculates equispaced grid points along a grid line  
AVTAN2 performs average tangents interpolation between two data points  
WEIGHT generates weighting values associated with interpolated data points.

In addition the following standard FORTRAN functions are called:

FLOAT converts integer to floating point  
IFIX integer value nearer to zero  
ABS absolute value  
ATAN arc tangent  
COS cosine

#### A.5 Input/output

The contour data being used by the program is required to be read twice. It is input to the program via FORTRAN channels 5 and 7. The output data is written to an output file via FORTRAN channel 6.

Appendix BSUBROUTINE SPECIFICATIONS FOR PROGRAM MAPGRIDSUBROUTINE CHECK

- Summary - This subroutine ensures that no two adjacent data points are coincident
- Subroutine statement - SUBROUTINE CHECK (F, G, IMAX, NMAX)
- Input argument - NMAX dimension for arrays F and G
- Input/output arguments - F } x and y co-ordinates of the data points  
G }  
IMAX number of data points
- Subordinate subprograms - None
- Explanation - A check is carried out with each pair of adjacent data points to determine whether they have the same co-ordinates. If they do one of the points is removed and IMAX is decremented by 1.

SUBROUTINE IDENT

- Summary - This subroutine modifies the format of contour data so that it is in a suitable form for subroutine COORDS
- Subroutine statement - SUBROUTINE IDENT (F, G, IMAX, NMAX, ITYPE)
- Input arguments - NMAX - dimension for arrays F and G  
ITYPE - contour type: open contour = +1, closed contour = 0
- Input/output arguments - F } x and y co-ordinates of data points defining the  
G } contour  
IMAX number of data points defining the contour
- Subordinate subprograms - None
- Explanation - See section 4.2.

SUBROUTINE COORDS

- Summary - This subroutine evaluates the co-ordinates at which y direction grid lines intersect a contour
- Subroutine statement - SUBROUTINE COORDS (F, G, H, X, Y, Z, IMAX, KMAX, WF, K)
- Input arguments - F } arrays containing the x and y co-ordinates of  
G } contour data points  
H contour height  
IMAX number of data points defining the contour  
KMAX dimensions of arrays X, Y and Z  
WF grid spacing
- Output arguments - X } arrays containing the x and y co-ordinates and the  
Y } height of intersection points between the grid lines  
Z } and the contour
- Input/output arguments - K On input : number of intersection points found by previous calls to the subroutine since the initialization of K in the main program  
On output : number of intersection points found by previous and present calls to the subroutine since the initialization of K in the main program

Subordinate subprograms - Function NFIX

## Subroutine INTERP

Explanation - Each pair of data points between 2 and IMAX-1 is taken in turn and the y direction grid lines which intersect the contour between these two points are derived. Subroutine INTERP is called to find the point of intersection of each of these grid lines with the contour. For each point of intersection K is incremented by 1 and the co-ordinates are entered into arrays X, Y and Z. It is possible that the contour is changing direction between the two data points with the result that one or more grid lines may cross the contour in two places (see Fig 17). In this situation, subroutine INTERP is called to determine whether the contour cuts the first grid line beyond the point farthest from the point at which the contour changes direction ( $P_{i-1}$  in Fig 17). If it does not, the next pair of points ( $P_i$  and  $P_{i+1}$ ) are considered. If it does, then the co-ordinates of the intersection points are entered into arrays X, Y and Z and K is suitably incremented. (If the contour just touches a grid line two points of intersection are still recorded although these points are coincident.) The next farther grid line is now taken then the process repeated until a grid line is found that does not cross the contour, when the next pair of points is then considered.

Only pairs of points between 2 and IMAX-1 are considered because of the nature of the interpolation algorithm used in subroutine INTERP.

A parameter EPS is defined in the subroutine, and is given as  $0.01 * WF$  in the program listing. This parameter should be chosen so that it is smaller than the digitizing step of the digitizer used to digitize the contour data.

FUNCTION NFIX

- Summary - This function gives the integer value of a number, the value being rounded down to the lower integer
- Function statement - FUNCTION NFIX(FF)
- Input argument - FF floating point number
- Output function - NFIX required integer value
- Subordinate subprograms - FORTRAN function IFIX
- Explanation - A call is made to a computer library function IFIX which gives the integer value of a number, the value being the nearer integer to zero. If the input value parameter has a value which is not an exact integer, the result of IFIX has ! subtracted from it if the input parameter to IFIX is negative. The user is advised that his compiler may have a library function equivalent to NFIX.

SUBROUTINE INTERP

- Summary - This subroutine calculates the co-ordinates at which a line X = XX intersects a contour between points (X2, Y2) and (X3, Y3)
- Subroutine statement - SUBROUTINE INTERP (X1, Y1, X2, Y2, X3, Y3, X4, Y4, XX, YY, YYS, ITEST)
- Input arguments - X1  
Y1  
X2  
Y2  
X3  
Y3  
X4  
Y4 } x and y co-ordinates of the points used by the interpolation algorithm
- XX x value at which an interpolated y value is required
- Output arguments - YY } interpolated y values  
YYS  
ITEST number of interpolated y values
- Subordinate subprograms - SUBROUTINE CUBIC

Explanation - The interpolated values of y are found using the average tangents algorithm described in section 3.3.3. The equation

$$x = a_x s^3 + b_x s^2 + c_x s + d_x$$

is solved by a call to subroutine CUBIC.

SUBROUTINE CUBICSummary

- This subroutine finds the roots of a third order polynomial in  $x$  in the range  $0 < x < x_{\max}$

Subroutine statement

- SUBROUTINE CUBIC (A, B, C, D, X1, X2, X3, XMAX, ITEST)

Input arguments

- A coefficient of the  $x^3$  term
- B coefficient of the  $x^2$  term
- C coefficient of the  $x$  term
- D constant term
- XMAX maximum value of  $x$  for which roots are required

Output arguments

- X1 first root
- X2 second root
- X3 third root
- ITEST number of roots

Subordinate subprograms - FORTRAN functions ABS  
ATAN  
COSExplanation

- This subroutine gives the real roots of the equation  $f(x) = ax^3 + bx^2 + cx + d = 0$ . A check is first carried out to determine whether the contribution of the  $x^3$  term is significant; the criteria adopted being that  $100ax_{\max} > b$ . If this criterion is not satisfied, the equation is treated as a quadratic. For this case a check is carried out to determine whether the contribution of the  $x^2$  term is significant; the criterion adopted being that  $100bx_{\max} > c$ . If this criterion is not satisfied, the polynomial is reduced to a linear equation. If  $100ax_{\max} > b$  but  $10^4ax_{\max}^2 < c$  the polynomial is again reduced to a linear equation. The roots of the equation for these three cases are as follows:

$$(1) \text{ Cubic equation } f(x) = ax^3 + bx^2 + cx + d = 0$$

$$Q = \frac{3c - b^2/a}{9a} \quad R = \frac{9bc/a - 27d - 2b^3/a^2}{54a} .$$

If  $Q^3 + R^2 > 0$ , then

$$x_1 = 2\sqrt{-Q} \cos \left[ \frac{1}{3} \cos^{-1} \left( \frac{R}{\sqrt{-Q^3}} \right) \right] - \frac{b}{3a} .$$

If  $Q^3 + R^2 > 0$

$$s = \sqrt[3]{R + \sqrt{Q^3 + R^2}} \quad t = \sqrt[3]{R - \sqrt{Q^3 + R^2}}$$

and

$$x_1 = s + t - \frac{b}{3a} .$$

Having found the first root the other two can now be found.

$$c' = x_1 + \frac{b}{a} \quad d' = x_1 \left( x_1 + \frac{b}{a} \right) + \frac{c}{a}$$

$$x_2 = \begin{cases} -\frac{c'}{2} - \sqrt{\left(\frac{c'}{2}\right)^2 - d'} & \text{if } -\frac{c'}{2} < 0 \\ -\frac{c'}{2} + \sqrt{\left(\frac{c'}{2}\right)^2 - d'} & \text{if } -\frac{c'}{2} \geq 0 \end{cases}$$

$x_2$  is real only if  $\left(\frac{c'}{2}\right)^2 - d' \geq 0$

$$x_3 = \frac{d'}{x_2} .$$

(2) Quadratic equation  $f(x) = bx^2 + cx + d = 0$

$$x_1, x_2 = -\frac{c \pm \sqrt{c^2 - 4bd}}{2b} .$$

The roots are real only if  $c^2 - 4bd \geq 0$ .

(3) Linear equation  $f(x) = cx + d = 0$

$$x_1 = -\frac{d}{c} .$$

Each of the roots is tested to determine whether it lies in the range  $0 < x < x_{max}$ . ITEST indicates the number of roots that satisfy this criteria. If there is only one root it is returned by X1 and if there are two they are returned by X1 and X2.

SUBROUTINE HEIGHTSummary

- This subroutine sorts the contour values into ascending order. The first and last values are contour levels within which interpolated values are to be restricted.

Subroutine statement

- SUBROUTINE HEIGHT (H, M, IH, HMIN, HMAX)

Input arguments

- M      number of contours used in program +2  
 HMIN    minimum allowed value of map data  
 HMAX    maximum allowed value of map data

Input/output arguments

- H      On input : array containing all the height values of all the contours used in the program  
 On output : array containing the values of all the different contour heights plus HMIN and HMAX  
 IH      On input : number of contours used in the program  
 On output : number of different contour heights +2

Subordinate subprograms

- None

Explanation

- HMIN and HMAX are entered into array H and then the values of H are sorted into ascending order using a bubble sort (see explanation under SUBROUTINE ORDER). A test is then carried out with each pair of values of H to see if any are equal. If the two values are found to be equal, one of the values is removed from the array and IH, denoting the values of interest in the array, is decremented by 1.

SUBROUTINE SELECTSummary

- This subroutine selects data points with a given x co-ordinate

Subroutine statement

- SUBROUTINE SELECT (X, Y, Z, KMAX, U, V, W, MAX, J, WW)

Input arguments

- X      arrays containing the x, y and z co-ordinates of the intersection points between grid lines and contours  
 Y      {  
 Z      }  
 KMAX    number of points of intersection  
 U      value of x for which data points are required  
 MAX    dimensions for arrays V and W  
 WW    grid spacing

Output arguments

- V      array containing y values with the x value U  
 W      array containing z values with the x value U  
 J      number of data points with x value U

Subordinate subprograms

- None

Explanation

- Each point of intersection is tested to determine whether the x value is equal to U. If it is, J is incremented by 1 and the corresponding y and z values are entered into arrays V and W respectively.

SUBROUTINE ORDER

Summary - This subroutine orders values in an array into ascending order

Subroutine statement - SUBROUTINE ORDER (V, W, LMAX)

Input argument - LMAX number of values to be sorted

Input/output arguments - V array of values to be sorted  
W array of values associated with V

Subordinate subprograms - None

Explanation - The method used is the bubble sort. This involves taking each pair of values of V in turn and checking to determine whether or not they are in the correct order. If they are not they are interchanged (together with the associated values of W) and a counter J is incremented. If J is non-zero (indicating that at least one pair of values has been interchanged) the process is repeated. When J is zero all the values are in the correct order.

SUBROUTINE CHECK2

Summary - This subroutine ensures that in the situation where a grid line just touches a contour, the two data points representing the points of intersection of the grid line with the contour are not coincident.

Subroutine statement - SUBROUTINE CHECK2 (Y, LMAX, WW)

Input arguments - LMAX number of data points  
WW grid spacing

Input/output arguments - Y array containing y values of data points

Subordinate subprograms - FORTRAN function ABS

Explanation - If a grid line just touches a contour subroutine COORDS outputs two points with the same co-ordinates, rather than one point, which will cause subsequent interpolation algorithms to fail unless corrected. This subroutine checks each adjacent pair of points to determine whether they have the same y values. If they do the y values of these points,  $y_{i-1}$  and  $y_i$ , are then replaced by  $y_{i-1} - E_1$  and  $y_i + E_2$  where

$$E_1 = |y_{i-1} - y_{i-2}|/5 \text{ unless } E_1 > E \text{ or } y_{i-2} \text{ does not exist, when } E_1 = E$$

$$E_2 = |y_{i+1} - y_i|/5 \text{ unless } E_2 > E \text{ or } y_{i+1} \text{ does not exist, when } E_2 = E$$

and  $E = 0.01 \times WW$ .

SUBROUTINE AVTANSummary

- This subroutine calculates equispaced data points along a grid line

Subroutine statement

- SUBROUTINE AVTAN (A, Y, Z, N, YY, ZZ, JDIM, H, IH, FRACTN, IROUTE, ZE)

Input arguments

- Y array containing y values of data points
- Z array containing z values of data points
- N number of data points
- YY { array containing y values at which interpolated values of z are required
- JDIM number of interpolated values required
- H IH FRACTN { used in SUBROUTINE AVTAN2 (see subroutine AVTAN2)
- IROUTE routing parameter (see explanation)
- ZE value of z for points outside the contour map

Output argument

- ZZ { array containing interpolated values corresponding to YY
- A array containing the average tangent at each of the data points

Subordinate subprograms - SUBROUTINE AVTAN2

Explanation - The average tangents at each of the data points ( $y_i$ ,  $z_i$ ) are first calculated. For each value of y at which interpolated values of z are required (yy) the data points are found which satisfy the criterion  $y_i < yy < y_{i+1}$ . A call is then made to subroutine AVTAN2 to obtain the interpolated value of z at yy. If the value of yy lies outside the range of the data points then one of two alternatives is taken depending upon the value of the routing parameter IROUTE.

IRROUTE ≠ 1 The interpolated value is set to the value of z corresponding to the y value closest to yy .

IRROUTE = 1 The interpolated value is set to ZE , a value specified by the user in the main program.

It should be noted that in arrays YY and Y the values must be arranged in ascending order and no two values of Y must be the same. The latter condition is ensured by a call to subroutine CHECK2 prior to the call to this subroutine.

SUBROUTINE AVTAN2Summary

- This subroutine performs average tangents interpolation between two data points

Subroutine statement

- SUBROUTINE AVTAN2 (Y1, Y2, Z1, Z2, A1, A2, YY, ZZ, H, IH, FRACTN)

Input arguments

- Y1      { y and z values and average tangent at the first  
Z1      } data point  
A1      }
- Y2      { y and z values and average tangent at the second  
Z2      } data point  
A2      }
- YY      { y value at which an interpolated z value is  
              required
- H      { array containing the values of all the different  
              contour heights plus minimum and maximum allowed  
              values of map data
- IH      number of values in H
- FRACTN set in main program, normally to 0.99

Output argument

- ZZ      interpolated z value

Subordinate subprograms

- None

Explanation

- The interpolated value of z is found using the average tangents algorithm described in section 3.3.1. The algorithm has been modified (see section 3.3.2) so that at turning points the interpolated value is not allowed to reach the next contour level and in other circumstances is constrained to remain between Z1 and Z2 .

SUBROUTINE WEIGHTSummary

- This subroutine generates weighting values associated with the interpolated data points along a grid line

Subroutine statement

- SUBROUTINE WEIGHT (Y, YY, W, N, JDIM, Z, A)

Input arguments

- Y array containing y values of data points  
 YY array containing y values of interpolated data points  
 N number of data points  
 JDIM number of interpolated data points  
 Z array containing z values of data points  
 A {array containing the average tangents at each of the  
 data points}

Output argument

- W array containing weighting values

Subordinate subprograms

- None

Explanation - for each interpolated point (yy) data points are found which satisfy the criterion  $y_i < yy < y_{i+1}$ . The weighting value associated with yy is given by:

$$W = \frac{1}{|y_{i+1} - y_i|}$$

If  $y_1 = yy$   $W = \frac{1}{|y_2 - y_1|}$

If  $A_i = A_{i+1} = 0$  and  $Z_i = Z_{i+1}$   $W = -1$

If yy lies outside the range of y  $W = 0$ .

It should be noted that the weighting value is not important when  $yy = y_i$  or  $y_{i+1}$ .

Appendix CLISTING OF PROGRAM MAPGRID

C \*\*\*\*\* PROGRAM MAPGRID \*\*\*\*\*

PAGE 0001

C \*\*\*\*\* PROGRAM MAPGRID \*\*\*\*\*

C \*\*\*\*  
C THIS PROGRAM CONVERTS A CONTOUR MAP INTO A UNIFORMLY SPACED GRID  
C OF POINTS

C CONTOUR FITTING ALGORITHM: AVERAGE TANGENTS

C \*\*\*\*  
CDIMENSION F(58),G(58),X(1000),Y(1000),Z(1000),R(58),S(58)  
DIMENSION U(59),V(59),T(59),WT(59),W(58),H(58)  
DIMENSION AA(59,59),WA(59,59)C  
C INITIALIZATION OF PARAMETERSMAX=58  
KMAX=1000  
IDIM=59  
JDIM=59  
IRROUTE=1  
ZE=8.0  
WW=1.0  
XST=-29.0  
YST=-29.0  
HMIN=8.0  
HMAX=250.0  
FRACTN=0.99  
DO 10 I=1, IDIM  
10 U(I)=FLOAT(I-1)\*WW  
DO 20 J=1, JDIM  
20 V(J)=FLOAT(J-1)\*WWC  
C GENERATE GRID IN F DIRECTIONC  
M=8  
KREC=8  
210 M=M+1  
READ (5,51,END=230) H(M),IMAX,ITYPE  
DO 220 I=1,IMAX  
READ (5,52) F(I),G(I)  
F(I)=F(I)-XST  
220 G(I)=G(I)-YST  
NMAX=IMAX  
CALL CHECK (F,G,IMAX,NMAX)  
NMAX=IMAX+2  
CALL IDENT (F,G,IMAX,NMAX,ITYPE)  
CALL COORDS (F,G,H(M),X,Y,Z,IMAX,KMAX,WW,KREC)  
GO TO 210230 M=M+1  
CALL HEIGHT (H,M,IM,HMIN,HMAX)  
DO 240 I=1, IDIM  
CALL SELECT (X,Y,Z,KREC,U(I),S,W,MAX,LMAX,WW)  
IF (LMAX LT. 1) GO TO 260  
CALL ORDER (S,W,LMAX)  
CALL CHECK2 (S,LMAX,WW)  
CALL AVTAN (A,S,W,LMAX,V,T,JDIM,H,IM,FRACTN,IRROUTE,ZE)  
CALL WEIGHT (S,V,WT,LMAX,JDIM,W,A)  
GO TO 250

C \*\*\*\*\* PROGRAM MAPGRID \*\*\*\*\*

PAGE 0002

```

260 DO 270 J=1,JDIM
  UT(J)=-2.0
270 T(J)=ZE
280 DO 290 J=1,JDIM
  AAC(I,J)=T(J)
290 WA(I,J)=UT(J)

C
C   GENERATE GRID IN G DIRECTION
C
  KREC=0
310 READ (7,51,END=330) HH,IMAX,ITYPE
  DO 320 I=1,IMAX
    READ (7,52) F(I),G(I)
    F(I)=F(I)-NST
320 G(I)=G(I)-YST
  IMAX=IMAX
  CALL CHECK (F,G,IMAX,MMAX)
  MMAX=IMAX+2
  CALL IDENT (G,F,IMAX,MMAX,ITYPE)
  CALL COORDS (G,F,HH,X,Y,Z,IMAX,KMAX,WW,KREC)
  GO TO 310
330 WRITE (6,53) IDIM,JDIM
  DO 340 J=1,JDIM
    CALL SELECT (X,Y,Z,KREC,V(J),S,W,MAX,LMAX,WW)
    IF (LMAX.LT.1) GO TO 360
    CALL ORDER (S,W,LMAX)
    CALL CHECK2 (S,LMAX,WW)
    CALL AVTAH (A,S,W,LMAX,U,T,IDLIM,H,IH,FRACTN,IRROUTE,ZE)
    CALL WEIGHT (S,U,WT,LMAX,IDLIM,W,A)
    GO TO 350
340 DO 370 I=1,IDLIM
  UT(I)=-2.0
370 T(I)=ZE
350 DO 340 I=1,IDLIM
  IF (WA(I,J).EQ.WT(I)) AAC(I,J)=(AAC(I,J)+T(I))/2.0
  IF (WA(I,J).EQ.WT(I)) GO TO 340
  IF (IRROUTE.NE.1) GO TO 345
  IF (WA(I,J).EQ.-2.0) GO TO 340
  IF (WT(I).EQ.-2.0) AAC(I,J)=T(I)
  IF (WT(I).EQ.-2.0) GO TO 340
  IF (WA(I,J).EQ.0.0) GO TO 340
  IF (WT(I).EQ.0.0) AAC(I,J)=T(I)
  IF (WT(I).EQ.0.0) GO TO 340
  IF (WT(I).GT.WA(I,J)) AAC(I,J)=T(I)
  GO TO 340
345 IF (WA(I,J).EQ.-1.0.AND.WT(I).LE.0.0) GO TO 340
  IF (WT(I).EQ.-1.0.AND.WA(I,J).LE.0.0) GO TO 346
  IF (WT(I).GT.WA(I,J)) AAC(I,J)=T(I)
  GO TO 340
346 AAC(I,J)=T(I)
340 CONTINUE

C
C   OUTPUT FINAL GRID
C
  DO 480 J=1,JDIM
480 WRITE (6,54) (AAC(I,JDIM+1-J),I=1,IDLIM)

C
C   INPUT/OUTPUT FORMATS
C
  S1 FORMAT (F6.1,IS,12)

```

C \*\*\*\*\* PROGRAM MAPGRID \*\*\*\*\* PAGE 0003

```

52 FORMAT (2F8.3)
53 FORMAT (I3,1X,I3)
54 FORMAT (12F6.1)

C
      STOP
      END
      SUBROUTINE CHECK (F,G,IMAX,NMAX)
C
C      THIS SUBROUTINE ENSURES THAT NO TWO ADJACENT DATA POINTS ARE COINCIDENT
C
      DIMENSION F(NMAX),G(NMAX)
      I=1
10    I=I+1
      IF (I.GT.IMAX) RETURN
      IF (F(I).NE.F(I-1)) GO TO 10
      IF (G(I).NE.G(I-1)) GO TO 10
      DO 20 J=I,IMAX
      F(J-1)=F(J)
20    G(J-1)=G(J)
      IMAX=IMAX-1
      I=I-1
      GO TO 10
      END
      SUBROUTINE IDENT (F,G,IMAX,NMAX,ITYPE)
C
C      THIS SUBROUTINE MODIFIES THE INPUT DATA FOR SUBROUTINE COORDS DEPENDING
C      UPON THE CONTOUR IDENTIFICATION GIVEN BY ITYPE
C      ITYPE=+1  OPEN CONTOUR
C      ITYPE= 0  CLOSED CONTOUR
C      ITYPE=-1  UNDEFINED
C
      DIMENSION F(NMAX),G(NMAX)
      IF (ITYPE) 10,20,30
10    RETURN
20    F(IMAX+1)=F(2)
      G(IMAX+1)=G(2)
      F(IMAX+2)=F(3)
      G(IMAX+2)=G(3)
      IMAX=IMAX+2
      RETURN
30    F(IMAX+1)=2.0*F(IMAX)-F(IMAX-1)
      G(IMAX+1)=2.0*G(IMAX)-G(IMAX-1)
      IMAX=IMAX+2
      DO 40 I=2,IMAX
      F(IMAX+2-I)=F(IMAX+1-I)
40    G(IMAX+2-I)=G(IMAX+1-I)
      F(1)=2.0*F(2)-F(3)
      G(1)=2.0*G(2)-G(3)
      RETURN
      END
      SUBROUTINE COORDS (F,G,H,X,Y,Z,IMAX,KMAX,WF,K)
C
C      THIS SUBROUTINE EVALUATES THE CO-ORDINATES AT WHICH THE GRID LINES
C      INTERSECT A CONTOUR. AVERAGE TANGENTS INTERPOLATION IS EMPLOYED.
C
      DIMENSION F(IMAX),G(IMAX),X(KMAX),Y(KMAX),Z(KMAX)
      EPS=ABS(0.01*WF)
      I=3
      IF1=NFIIX(F(I-1)/WF)
      IF (ABS(FLDAT(IF1)*WF-F(I-1)).GT.EPS) GO TO 5

```

PAGE 8884

C \*\*\*\*\* PROGRAM MAPGRID \*\*\*\*\*

```

K=K+1
X(K)=F(I)
Y(K)=G(I)
Z(K)=H
5 I=I-1
80 I=I+1
IF (I.GT.IMAX-1) RETURN
IF1=NFIK(F(I-1)/WF)
IF2=NFIK(F(I)/WF)
FF=(F(I+1)-F(I))*(F(I-1)-F(I-2))
IF (FF.LT.0.0) GO TO 50
NF=IF2-IF1
IF (NF) 40,50,60
60 DO 20 N=1,NF
K=K+1
X(K)=FLOAT(IF1)*WF+FLOAT(N)*WF
IF (ABS(X(K)-F(I)).GT.EPS) GO TO 10
Y(K)=G(I)
GO TO 20
10 CALL INTERP(F(I-2),G(I-2),F(I-1),G(I-1),F(I),G(I),F(I+1)
     ,G(I+1),X(K),Y(K),YY1,ITEST)
20 Z(K)=H
GO TO 80
40 NF=-NF
IF (ABS(FLOAT(IF2)*WF-F(I)).LT.EPS) NF=NF+1
DO 30 N=1,NF
XX=FLOAT(IF1)*WF-FLOAT(N-1)*WF
IF (ABS(XX-F(I-1)).LT.EPS) GO TO 30
K=K+1
X(K)=XX
IF (ABS(X(K)-F(I)).GT.EPS) GO TO 70
Y(K)=G(I)
Z(K)=H
GO TO 30
70 CALL INTERP(F(I-2),G(I-2),F(I-1),G(I-1),F(I),G(I),F(I+1)
     ,G(I+1),X(K),Y(K),YY1,ITEST)
Z(K)=H
30 CONTINUE
GO TO 80
50 XX=FLOAT(IF1)*WF
IF (ABS(XX-F(I)).GT.EPS) GO TO 55
K=K+1
X(K)=F(I)
Y(K)=G(I)
Z(K)=H
55 IF (FF.GE.0.0) GO TO 80
N=0
ISIGN=1
IF ((F(I+1)+F(I-2))-(F(I)+F(I-1))).LT.0.0 ISIGN=-1
90 ISIGN=-1
N=1
IF (IF2.GT.IF1) IF1=IF2
GO TO 110
100 IF (IF2.LT.IF1) IF1=IF2
110 N=N+1*ISIGN
XX=FLOAT(IF1+N)*WF
CALL INTERP(F(I-2),G(I-2),F(I-1),G(I-1),F(I),G(I),F(I+1)
     ,G(I+1),XX,YY1,YY2,ITEST)
IF (ITEST.EQ.0) GO TO 80
K=K+1

```

PAGE 8885

C \*\*\*\*\* PROGRAM MAPGRID \*\*\*\*\*

```

X(K)=XX
Y(K)=YY1
Z(K)=H
IF (ITEST.EQ.1) GO TO 110
K=K+1
X(K)=XX
Y(K)=YY2
Z(K)=H
GO TO 110
END
FUNCTION NFIX(FF)
C
C THIS FUNCTION GIVES THE INTEGER VALUE OF A NUMBER, THE VALUE BEING
C ROUNDED DOWN TO THE LOWER INTEGER.
C
NFIIX=IFIX(FF)
IF (FLOAT(NFIIX) EQ FF) RETURN
IF (FF.LT.0.0) NFIIX=NFIIX-1
RETURN
END
SUBROUTINE INTERP (X1,Y1,X2,Y2,X3,Y3,X4,Y4,XX,YY,YY5,ITEST)
C
C THIS SUBROUTINE CALCULATES THE CO-ORDINATES AT WHICH A LINE X=XX
C INTERSECTS A CONTOUR BETWEEN POINTS (X2,Y2) AND (X3,Y3)
C
D1=((X2-X1)**2+(Y2-Y1)**2)**0.5
D2=((X3-X2)**2+(Y3-Y2)**2)**0.5
D3=((X4-X3)**2+(Y4-Y3)**2)**0.5
AX1=((X2-X1)/D1+(X3-X2)/D2)/2.0
AX2=((X3-X2)/D2+(X4-X3)/D3)/2.0
AY1=((Y2-Y1)/D1+(Y3-Y2)/D2)/2.0
AY2=((Y3-Y2)/D2+(Y4-Y3)/D3)/2.0
DX=X3-X2
DY=Y3-Y2
AX=(AX1*D2+AX2*D2-2.0*DX)/D2**3
BX=(3.0*DX-AX2*D2-2.0*AX1*D2)/D2**2
CX=AX1
AY=(AY1*D2+AY2*D2-2.0*DY)/D2**3
BY=(3.0*DY-AY2*D2-2.0*AY1*D2)/D2**2
CY=AY1
DX=X2-XX
CALL CUBIC (AX,BX,CX,DX,DS1,DS2,DS3,D2,ITEST)
IF (ITEST.EQ.0) RETURN
YY=AY+DS1**3+BY*DS1**2+CY*DS1+Y2
IF (ITEST.EQ.1) RETURN
YY5=AY*DS2**3+BY*DS2**2+CY*DS2+Y2
ITEST=2
RETURN
END
SUBROUTINE CUBIC (A,B,C,D,X1,X2,X3,XMAX,ITEST)
C
C THIS SUBROUTINE FINDS THE ROOTS OF A THIRD ORDER POLYNOMIAL IN THE
C RANGE 0.0.LT.X LT.XMAX
C
IF (ABS(A*XMAX**100.0).LT.ABS(B)) GO TO 38
IF (ABS(A*XMAX**2+10000.0) LT ABS(C)) GO TO 38
C
C SOLVE THIRD ORDER POLYNOMIAL
C
Q=(3.0*C-(B**2)/A)/(9.0*A)

```

PAGE 0006

C \*\*\*\*\* PROGRAM MAPGRID \*\*\*\*\*

```

R=(9.0+B*C/A-27.0*D-2.0*B**3/A**2)/(54.0*A)
IF (Q**3+R**2.LT.0.0) GO TO 5
S=R+(Q**3+R**2)**0.5
SIGN=1.0
IF (S.LT.0.0) SIGN=-1.0
S=SIGN*(ABS(S))**((1.0/3.0))
T=R-(Q**3+R**2)**0.5
SIGN=1.0
IF (T.LT.0.0) SIGN=-1.0
T=SIGN*(ABS(T))**((1.0/3.0))
X1=S+T-B/(3.0*A)
GO TO 10
5 DUM=ATAN((ABS(-Q**3/R**2-1.0))**0.5)
IF (R.LT.0.0) DUM=3.141592654-DUM
X1=2.0*(-Q)**0.5*COS(DUM/3.0)-B/(3.0*A)
10 ITEST=1
BB=X1+B/A
CC=X1*(X1+B/A)+C/A
TEST=(BB/2.0)**2-CC
IF (TEST.LT.0.0) GO TO 25
IF (-BB/2.0.LT.0.0) GO TO 15
X3=-BB/2.0+TEST**0.5
GO TO 20
15 X3=-BB/2.0-TEST**0.5
20 X2=CC/X3
ITEST=3
25 GO TO 45
C
C   SOLVE SECOND ORDER POLYNOMIAL
C
30 IF (ABS(B*XMAX+100.0).LT.ABS(C)) GO TO 48
TEST=C**2-4.0*B*D
IF (TEST.GE.0.0) GO TO 35
ITEST=0
RETURN
35 X1=(-C-TEST**0.5)/(2.0*B)
X2=(-C+TEST**0.5)/(2.0*B)
ITEST=2
GO TO 45
C
C   SOLVE FIRST ORDER POLYNOMIAL
C
40 X1=-D/C
ITEST=1
C
C   WHICH ROOTS LIE IN THE RANGE 0.0.LT.X.LT.XMAX
C
45 IF (X1.GT.0.0.AND.X1.LT.XMAX) GO TO 65
ITEST=ITEST-1
IF (ITEST-1) 50,55,60
50 RETURN
55 X1=X2
GO TO 45
60 X1=X2
X2=X3
GO TO 45
65 IF (ITEST.EQ.1) RETURN
70 IF (X2.GT.0.0.AND.X2.LT.XMAX) GO TO 75
ITEST=ITEST-1
IF (ITEST.EQ.1) RETURN

```

```

C      ***** PROGRAM MAPGRID *****          PAGE 8887

X2=X3
GO TO 70
75 IF (ITEST.EQ.2) RETURN
IF (X3.GT.0.0.AND.X3.LT.XMAX) RETURN
ITEST=ITEST-1
RETURN
END
SUBROUTINE HEIGHT (H,M,IH,HMIN,HMAX)

C THIS SUBROUTINE SORTS THE CONTOUR VALUES INTO ASCENDING ORDER. THE FIRST
C AND LAST VALUES ARE CONTOUR LEVELS WITHIN WHICH INTERPOLATED VALUES ARE
C TO BE RESTRICTED
C
DIMENSION H(M)
H(M-1)=HMIN
H(M)=HMAX
LMAX=M-1
20 J=0
DO 10 I=1,LMAX
IF (H(I+1).GE.H(I)) GO TO 10
J=J+1
DUM=H(I+1)
H(I+1)=H(I)
H(I)=DUM
10 CONTINUE
IF (J.NE.0) GO TO 20
I=0
40 I=I+1
35 IF (I.GT.LMAX) GO TO 38
IF (H(I+1).NE.H(I)) GO TO 40
DO 50 J=I,LMAX
50 H(J)=H(J+1)
LMAX=LMAX-1
GO TO 35
38 IH=LMAX+1
RETURN
END
SUBROUTINE SELECT (X,Y,Z,KMAX,U,V,W,MAX,J,WW)

C THIS SUBROUTINE SELECTS DATAPoints WITH A GIVEN X CO-ORDINATE
C
DIMENSION X(KMAX),Y(KMAX),Z(KMAX),V(MAX),W(MAX)
I=0
J=0
EPS=ABS(0.1*WW)
10 I=I+1
IF (I.GT.KMAX) GO TO 20
IF (ABS(X(I)-U).GT.EPS) GO TO 10
J=J+1
V(J)=Y(I)
W(J)=Z(I)
GO TO 10
20 RETURN
END
SUBROUTINE ORDER (V,W,LMAX)

C THIS SUBROUTINE ORDERS THE VALUES OF V INTO ASCENDING ORDER
C
DIMENSION V(LMAX),W(LMAX)
IF (LMAX.EQ.1) RETURN

```

PAGE 8000

C \*\*\*\*\* PROGRAM MAPGRID \*\*\*\*\*

```

LLMAX=LMAX-1
20 J=0
DO 10 I=1,LLMAX
IF (V(I+1).GE.V(I)) GO TO 10
J=J+1
DUM=V(I+1)
V(I+1)=V(I)
V(I)=DUM
DUM=U(I+1)
U(I+1)=U(I)
U(I)=DUM
10 CONTINUE
IF (J.NE.0) GO TO 20
RETURN
END
SUBROUTINE CHECK2 (Y,LMAX,UW)

C THIS SUBROUTINE ENSURES THAT IN THE SITUATION WHERE A GRID LINE JUST
C TOUCHES A CONTOUR THE TWO DATA POINTS REPRESENTING THE POINTS OF
C INTERSECTION OF THE GRID LINE WITH THE CONTOUR ARE NOT COINCIDENT
C
DIMENSION Y(LMAX)
IF (LMAX.LT.2) RETURN
EPS=0.01*UW
I=1
10 I=I+1
IF (I.GT.LMAX) RETURN
IF (Y(I).NE.Y(I-1)) GO TO 10
IF (I.GT.2) YL=Y(I-2)
IF (I.LT.LMAX) YM=Y(I+1)
IF (I.EQ.2) YL=Y(I-1)-5.0*EPS
IF (I.EQ.LMAX) YM=Y(I)+5.0*EPS
SS=ABS(YL-Y(I-1))/5.0
IF (SS.GT.EPS) SS=EPS
Y(I-1)=((YL-Y(I-1))/ABS(YL-Y(I-1))*SS)+Y(I-1)
SS=ABS(YM-Y(I))/5.0
IF (SS.GT.EPS) SS=EPS
Y(I)=((YM-Y(I))/ABS(YM-Y(I)))*SS+Y(I)
GO TO 10
END
SUBROUTINE AVTAH (A,Y,Z,H,YY,ZZ,JDIM,H,IH,FRACTN,IRROUTE,ZE)

C THIS SUBROUTINE CALCULATES EQUISPACED DATA POINTS ALONG A GRID LINE
C
DIMENSION A(H),YY(JDIM),ZZ(JDIM),Y(H),Z(H),H(IH)
E1=Z(1)
E2=Z(H)
IF (IRROUTE.NE.1) GO TO 5
E1=ZE
E2=ZE
5 IF (H.EQ.1) GO TO 70
A(1)=(Z(2)-Z(1))/(Y(2)-Y(1))
A(H)=(Z(H)-Z(H-1))/(Y(H)-Y(H-1))
IF (H.EQ.2) GO TO 15
INT=H-1
DO 10 I=2,INT
A1=(Z(I)-Z(I-1))/(Y(I)-Y(I-1))
A2=(Z(I+1)-Z(I))/(Y(I+1)-Y(I))
10 A(I)=(A1+A2)/2.0
15 J=0

```

PAGE 0009

C \*\*\*\*\* PROGRAM MAPGRID \*\*\*\*\*

```

I=1
20 J=J+1
IF (J.GT.JDIM) RETURN
IF (YY(J)-Y(I)) 25,30,30
25 ZZ(J)=E1
GO TO 20
30 J=J+1
IF (J.GT.JDIM) RETURN
30 IF (YY(J)-Y(I)) 20,40,45
40 ZZ(J)=Z(I)
GO TO 35
45 IF (YY(J)-Y(I+1)) 55,60,65
50 I=I+1
IF (I+1.GT.N) GO TO 50
GO TO 30
50 ZZ(J)=Z(I+1)
GO TO 35
55 CALL AVTAN2 (Y(I),Y(I+1),Z(I),Z(I+1),A(I),A(I+1),YY(J),ZZ(J),
H,IH,FRACTH)
GO TO 35
70 J=1
50 DO 75 L=J,JDIM
75 ZZ(L)=E2
RETURN
END
SUBROUTINE AVTAN2 (Y1,Y2,Z1,Z2,A1,A2,YY,ZZ,H,IH,FRACTH)

C
C THIS SUBROUTINE IS THE AVERAGE TANGENTS ALGORITHM FOR CONTOUR CUT DATA.
C IT ENSURES THAT INTERPOLATED VALUES REMAIN WITHIN THE CONTOUR INTERVAL
C UNDER CONSIDERATION OR FOR TURNING POINTS DOES NOT DEVIATE FROM THE
C INITIAL CONTOUR DATA BY MORE THAN ONE CONTOUR INTERVAL
C
DIMENSION H(IH)
DUM1=A1
DUM2=A2
DUM3=Z1
DUM4=Z2
DUM5=YY
IF (Z2.GE.Z1) GO TO 5
DUM=Z1
Z1=Z2
Z2=DUM
DUM=A1
A1=-A2
A2=-DUM
YY=Y2+Y1-YY
5 DY=Y2-Y1
DZ=Z2-Z1
A=(A1*DY+A2*DZ-2.0*DZ)/(DY**3)
B=(3.0*DZ-2.0*A1*DY-A2*DY)/(DY**2)
C=A1
Y=YY-Y1
ZZ=A*YY**3+B*YY**2+C*Y+Z1
IF (Z2.NE.Z1) GO TO 10
IF (A2.EQ.A1) GO TO 70
10 ITEST=0
TEST=B**2-3.0*A*C
IF (ABS(A*DY+150.0).GT.ABS(B)) GO TO 15
YS1=-C/(2.0*B)
IF (YS1.LT.0.0.OR.YS1.GT.DY) GO TO 70

```

C \*\*\*\*\* PROGRAM MAPGRID \*\*\*\*\*

```

ITEST=1
GO TO 35
15 IF (TEST) 70,20,25
20 YS1=-B/(3.0*A)
IF (YS1.LT.0.0.OR.YS1.GT.DY) GO TO 70
ITEST=1
GO TO 35
25 YS1=(-B-TEST**B.5)/(3.0*A)
IF (YS1.LT.0.0.OR.YS1.GT.DY) GO TO 30
ITEST=1
30 YS2=(-B+TEST**B.5)/(3.0*A)
IF (YS2.LT.0.0.OR.YS2.GT.DY) GO TO 35
IF (ITEST.EQ.0) YS1=YS2
ITEST=ITEST+1
IF (ITEST.NE.2) GO TO 35
IF (YS1.LT.YS2) GO TO 36
DUM=YS1
YS1=YS2
YS2=DUM
36 IF (A1.NE.0.0) GO TO 37
YS1=YS2
ITEST=1
IF (A2.NE.0.0) GO TO 35
ITEST=0
GO TO 35
37 IF (A2.NE.0.0) GO TO 35
ITEST=1
35 IF (ITEST.EQ.0) GO TO 70
IF (ITEST.EQ.2) GO TO 55
ISIGN=+1
IF (A1-A2.LT.0.0) ISIGN=-1
DO 40 I=1,IH
40 IF (Z1.EQ.H(I)) GO TO 45
45 IF (ISIGN.EQ.1) I=I+1
CONT=H(I)-H(I-1)
ZT=A*YS1**3+B*YS1**2+C*YS1
IF (ABS(ZT).LT.CONT) GO TO 70
DZ=FLOAT(ISIGN)*FRACTH*CONT
IF (Y.GT.YS1) GO TO 50
A=(A1*YS1-2.0*DZ)/YS1**3
B=(3.0*DZ-2.0*A1*YS1)/YS1**2
ZZ=A*Y**3+B*Y**2+C*Y+Z1
IF (ABS(ZZ-Z1).GT.ABS(DZ)) ZZ=Z1+DZ
GO TO 70
50 DY=DY-YS1
Y=Y-YS1
A=(A2*DY+2.0*DZ)/DY**3
B=(-3.0*DZ-A2*DY)/DY**2
ZZ=A*Y**3+B*Y**2+Z1+DZ
IF (ABS(ZZ-Z1).GT.ABS(DZ)) ZZ=Z1+DZ
GO TO 70
55 ZT1=A*YS1**3+B*YS1**2+C*YS1+Z1
ZT2=A*YS2**3+B*YS2**2+C*YS2+Z1
IF (ZT1.LT.ZZ.AND.ZT2.GT.Z1) GO TO 70
IF (Y.GT.YS1) GO TO 60
IF (ZT1.LT.ZZ) GO TO 70
DZ=FRACTH*(Z2-Z1)
A=(A1*YS1-2.0*DZ)/YS1**3
B=(3.0*DZ-2.0*A1*YS1)/YS1**2
ZZ=A*Y**3+B*Y**2+A1*Y+Z1

```

C \*\*\*\*\* PROGRAM MAPGRID \*\*\*\*\*

```

IF (Z2.GT.Z1+DZ) Z2=Z1+DZ
GO TO 70
60 IF (Y.GT.YS2) GO TO 65
IF (ZT1.GE.Z2) ZT1=Z1+FRACTH*(Z2-Z1)
IF (ZT2.LE.Z1) ZT2=Z2-FRACTN*(Z2-Z1)
DZ=ZT2-ZT1
DY=Y-S2-Y1
Y=Y-Y1
A=-2.B*DZ/DY**3
B=3.B*DZ/DY**2
Z2=A*Y**3+B*Y**2+ZT1
IF (Z2.GT.ZT1) Z2=ZT1
IF (Z2.LT.ZT2) Z2=ZT2
GO TO 70
65 IF (ZT2.GT.Z1) GO TO 70
DZ=FRACTH*(Z2-Z1)
DY=DY-Y2
Y=Y-Y2
A=(A2*DY-2.B*DZ)/DY**3
B=(3.B*DZ-A2*DY)/DY**2
Z2=A*Y**3+B*Y**2+Z2-DZ
IF (Z2.LT.Z2-DZ) Z2=Z2-DZ
70 A1=DUM1
A2=DUM2
Z1=DUM3
Z2=DUM4
YY=DUM5
RETURN
END
SUBROUTINE WEIGHT (Y,YY,W,N,JDIM,Z,A)
C
C THIS SUBROUTINE GENERATES WEIGHTING VALUES ASSOCIATED WITH THE
C INTERPOLATED DATA POINTS ALONG A GRID LINE
C
DIMENSION YY(JDIM),W(JDIM),Y(N),Z(N),A(N)
J=1
IF (N.EQ.1) GO TO 70
J=2
I=1
10 J=J+1
IF (J.GT.JDIM) RETURN
IF (YY(J)-Y(I)).GT.30,30,30
20 W(J)=0.0
GO TO 10
40 J=J+1
IF (J.GT.JDIM) RETURN
30 IF (YY(J)-Y(I+1)).GT.50,50,60
60 I=I+1
IF (I+1.GT.N) GO TO 70
GO TO 30
50 W(J)=ABS(1.0/(Y(I+1)-Y(I)))
IF (Z(I).EQ.Z(I+1).AND.A(I).EQ.0.0.AND.A(I+1).EQ.0.0) W(J)=-1.0
GO TO 40
70 DO 80 L=J,JDIM
80 W(L)=0.0
RETURN
END

```

Appendix D

LINEAR INTERPOLATION VERSION OF SUBROUTINE COORDS

PAGE 0001

```
SUBROUTINE COORDS (F,G,H,X,Y,Z,IMAX,KMAX,WF,K,IEXIT)
C
C THIS SUBROUTINE EVALUATES THE CO-ORDINATES AT WHICH THE GRID LINES
C INTERSECT A CONTOUR. LINEAR INTERPOLATION IS EMPLOYED.
C
DIMENSION F(IMAX),G(IMAX),X(KMAX),Y(KMAX),Z(KMAX)
EPS=ABS(0.01*WF)
I=3
IF1=HFIX(F(I-1)/WF)
IF (ABS(FLOAT(IF1)*WF-F(I-1)).GT.EPS) GO TO 5
K=K+1
X(K)=F(I)
Y(K)=G(I)
Z(K)=H
5 I=I-1
88 I=I+1
IF (I.GT.IMAX-1) RETURN
IF1=HFIX(F(I-1)/WF)
IF2=HFIX(F(I)/WF)
FF=(F(I+1)-F(I))*(F(I)-F(I-1))
NF=IF2-IF1
IF (NF) 40,50,60
60 DO 20 N=1,NF
K=K+1
X(K)=FLOAT(IF1)*WF+FLOAT(N)*WF
IF (ABS(X(K)-F(I)).GT.EPS) GO TO 10
Y(K)=G(I)
Z(K)=H
IF (FF) 110,80,80
10 Y(K)=((X(K)-F(I-1))*(G(I)-G(I-1))/(F(I)-F(I-1)))+G(I-1)
20 Z(K)=H
GO TO 88
40 NF=-NF
IF (ABS(FLOAT(IF2)*WF-F(I)).LT.EPS) NF=NF+1
DO 30 N=1,NF
XX=FLOAT(IF1)*WF-FLOAT(N-1)*WF
IF (ABS(XX-F(I-1)).LT.EPS) GO TO 30
K=K+1
X(K)=XX
IF (ABS(X(K)-F(I)).GT.EPS) GO TO 70
Y(K)=G(I)
Z(K)=H
IF (FF) 110,80,80
70 Y(K)=((X(K)-F(I-1))*(G(I)-G(I-1))/(F(I)-F(I-1)))+G(I-1)
30 Z(K)=H
GO TO 88
50 XX=FLOAT(IF1)*WF
IF (ABS(XX-F(I)).GT.EPS) GO TO 88
K=K+1
IF (K.GT.KMAX) GO TO 120
X(K)=F(I)
Y(K)=G(I)
Z(K)=H
IF (FF) 110,80,80
110 K=K+1
X(K)=X(K-1)
Y(K)=Y(K-1)
Z(K)=Z(K-1)
GO TO 88
END
```

Appendix EPROGRAMS TO CHECK OPERATION OF PROGRAM MAPGRID

E1

```

C      ***** PROGRAM DRIVER1 *****
C
C      THIS PROGRAM PRODUCES CONTOURS OF A 2-D GAUSSIAN, CENTRE (0,0)
C
C      DIMENSION X(13),Y(13)
C      ITYPE=0
C      INC=12
C      PI=3.141592654
C      HMAX=248.0
C      HPBW=16.0
C      CONT=25.0
C      CONST=ALOG(2.0)/HPBW**2
C      DO 10 J=1,9
C      F=CONT*FLOAT(10-J)
C      R=( ALOG(HMAX/F))/CONST)**0.5
C      DTHETA=2.0*PI/FLOAT(INC)
C      DO 20 I=1,INC
C      THETA=DTHETA*FLOAT(I-1)
C      X(I)=R*COS(THETA)
C      20 Y(I)=R*SIN(THETA)
C      INUM=INC+1
C      X(INUM)=X(1)
C      Y(INUM)=Y(1)
C      WRITE (6,61) F,INUM,ITYPE
C 61  FORMAT (F6.1,I5,I2)
C      WRITE (6,62) (X(I),Y(I),I=1,INUM)
C 62  FORMAT (2F8.3)
C      10 CONTINUE
C      STOP
C      END

```

E2

```

C      ***** PROGRAM DRIVER2 *****
C
C      THIS PROGRAM EVALUATES A 2-D GAUSSIAN OVER A 39x39 GRID
C
C      DIMENSION Z(39)
C      PI=3.141592654
C      HMAX=248.0
C      HPBW=16.0
C      CONST=ALOG(2.0)/HPBW**2
C      IDIM=39
C      JDIM=39
C      WRITE (6,68) IDIM,JDIM
C 68  FORMAT (I3,IX,I3)
C      DO 10 J=1,JDIM
C      Y=-29.0+FLOAT(J-1)
C      DO 20 I=1,IDIM
C      X=-29.0+FLOAT(I-1)
C      RR=X**2+Y**2
C      Z(I)=248.0/EXP(CONST*RR)
C 20  IF (Z(I) LT .25.0) Z(I)=25.0
C      10 WRITE (6,61) (Z(I),I=1,39)
C 61  FORMAT (12F6.1)
C      STOP
C      END

```

Appendix F  
PROGRAM EXPAND

F.1 Function

To expand (or contract) the number of points in a grid in either or both the x and y directions.

F.2 Dimensions of arrays

X	{	number of points in a row or column, whichever is greatest (IDIM)
A		
Z		

XX	{	number of points in a column after expansion (JJDIM)
ZZ		

AA IDIM, JJDIM.

F.3 Program units

Subroutine AVTAN : Calculates equispaced data points along a grid line. (This subroutine is very similar to that described in Appendix B. The main difference is that interpolated values outside the dataset are only set to the value of the nearest point in the dataset, no other option being available.)

Subroutine AVTAN2 : Average tangents algorithm for uniformly spaced data.

In addition the following FORTRAN functions are called:

IFIX : integer value nearer to zero  
FLOAT : converts integer to floating point.

F.4 Input/output

Input via FORTRAN channel 1 : expansion factors for x and y directions  
Input via FORTRAN channel 5 : the input grid (formatted as output by MAPGRID)  
Output via FORTRAN channel 6 : expanded grid (format similar to that of input grid).

F.5 Listing of program EXPAND

```

C
C      ***** PROGRAM EXPAND *****

C      THIS PROGRAM EXPANDS OR CONTRACTS THE NUMBER OF POINTS IN A GRID

C
COMMON X(257),XX(1855),A(257),Z(257),ZZ(1855),AA(257,1855)
COMMON AT(257,257)
READ (1,11) XM,YM
11 FORMAT (2FB.3)
READ (5,51) IDIM,JDIM
51 FORMAT (I3,1X,I3)
DO 18 J=1,JDIM
18 READ (5,52) (AT(I,J),I=1,1DIM)
52 FORMAT (12F6.1)
IPTS=IFIX(XM*FLOAT(IDIM-1))+1
JPTS=IFIX(YM*FLOAT(JDIM-1))+1
WRITE (6,61) IPTS,JPTS
61 FORMAT (I4,1X,I4)
SCALE=FLOAT(CDIM-1)/FLOAT(JPTS-1)
DO 28 J=1,JPTS
28 XX(J)=FLOAT(J-1)*SCALE
DO 38 J=1,JDIM
38 XC(J)=FLOAT(J-1)
DO 48 I=1,1DIM
DO 58 J=1,JDIM
58 Z(J)=AT(I,J)
CALL AVTAN (A,X,Z,JDIM,XX,ZZ,IPTS)
DO 48 J=1,JPTS
48 AC(I,J)=ZZ(J)
SCALE=FLOAT(IDIM-1)/FLOAT(IPTS-1)
DO 68 I=1,IPTS
68 XX(I)=FLOAT(I-1)*SCALE
DO 78 I=1,1DIM
78 XC(I)=FLOAT(I-1)
DO 88 J=1,JPTS
88 DO 98 I=1,1DIM
98 Z(I)=AC(I,J)
CALL AVTAN (A,X,Z,1DIM,XH,ZZ,IPTS)
88 WRITE (6,52) (ZZ(I),I=1,IPTS)
STOP
END
SUBROUTINE AVTAN (A,Y,Z,H,YY,ZZ,JDIM)

C      THIS SUBROUTINE CALCULATES EQUISPACED DATA POINTS ALONG A GRID LINE

C
DIMENSION A(N),YY(JDIM),ZZ(JDIM),Y(H),Z(N)
E1=Z(1)
E2=Z(H)
IF (H EQ 1) GO TO 78
AC(1)=(Z(2)-Z(1))/(Y(2)-Y(1))
AC(H)=(Z(H)-Z(H-1))/(Y(H)-Y(H-1))
IF (H LT 3) GO TO 15
INT=H-1
DO 18 I=2,INT
A1=(Z(I)-Z(I-1))/(Y(I)-Y(I-1))
A2=(Z(I+1)-Z(I))/(Y(I+1)-Y(I))
18 AC(I)=(A1+A2)/2.0
15 J=0
I=1
28 J=J+1

```

C

PAGE 8882

```

      IF (J.GT.JDIM) RETURN
      IF (YY(J)-Y(I)) 25,38,38
25  ZZ(J)=E1
     GO TO 28
35  J=J+1
     IF (J.GT.JDIM) RETURN
38  IF (YY(J)-Y(I)) 28,48,45
48  ZZ(J)=Z(I)
     GO TO 35
45  IF (YY(J)-Y(I+1)) 55,68,65
65  I=I+1
     IF (I+1.GT.N) GO TO 58
     GO TO 38
68  ZZ(J)=Z(I+1)
     GO TO 35
55  CALL AVTAN2 (A(I+1),A(I),YY(J),ZZ(J),Y(I+1),Y(I),Z(I+1),Z(I))
     GO TO 35
78  J=1
58  DO 75 L=J,JDIM
75  ZZ(L)=E2
     RETURN
    END
    SUBROUTINE AVTAN2 (A2,A1,YY,ZZ,Y2,Y1,Z2,Z1)
C
C   THIS SUBROUTINE IS THE AVERAGE TANGENTS ALGORITHM FOR EVENLY SPACED DATA.
C
      DY=Y2-Y1
      DZ=Z2-Z1
      A=(A1+DY+A2+DY-2*B*DZ)/(DY**3)
      B=(3*B*DZ-2*B*A1+DY-A2+DY)/(DY**2)
      C=A1
      Y=YY-Y1
      ZZ=A*Y**3+B*Y**2+C*Y+Z1
      RETURN
    END

```

Appendix G  
PROGRAM SECTION

G.1 Function

To produce a cross-section across a grid of points.

G.2 Dimensions of arrays

X	}	maximum number of points required in a cross-section
Y		
Z		
IW		

ZT      4  
ZZ      number of points in a row in the grid, 4.

G.3 Program units

Subroutine AVTAN : average tangents algorithm for uniformly spaced data.

In addition the following standard FORTRAN functions are called:

FLOAT : converts integer to floating point  
IFIX : integer value nearer to zero.

G.4 Input/output

Input via FORTRAN channel 1 : Co-ordinates of top left corner of grid and grid spacing.

Co-ordinates of the two end points of the required cross-section.

Number of points required in the cross-section.

Input via FORTRAN channel 5 : The input grid (formatted as output by MAPGRID).

Output via FORTRAN channel 6 : Number of points in the cross-section and the x and y co-ordinates and the height of the points in the cross-section.

G.5 Listing of program SECTION

PAGE 0001

C

```

C ***** PROGRAM SECTION *****
C
C DIMENSION Z2(1055,4),X(2000),Y(2000),Z(2000),ZT(4),IU(2000)
C
C INPUT CO-ORDINATES OF THE TOP LEFT HAND CORNER OF GRID AND THE GRID
C SPACING
C
C READ (1,11) X0,Y0,W
C
C INPUT THE CO-ORDINATES OF THE TWO ENDS OF THE REQUIRED CROSS-SECTION
C
C READ (1,12) X1,Y1,X2,Y2
C
C INPUT NUMBER OF POINTS REQUIRED IN THE CROSS-SECTION. IF THE NUMBER IS
C LESS THAN 2 THE PROGRAM CHOOSES A NUMBER BASED ON THE GRID SPACING
C
C READ (1,13) IPTS
C
C CHANGE DIRECTION OF CUT IF Y1.LT.Y2
C
C IDIRH=1
C IF (Y1.GE.Y2) GO TO 5
C DUM=X1
C X1=X2
C X2=DUM
C DUM=Y1
C Y1=Y2
C Y2=DUM
C IDIRH=-1
5 CONTINUE
C
C CALCULATE POINTS AT WHICH INTERPOLATED VALUES ARE REQUIRED
C
C IF (IPTS.GE.2) GO TO 10
C D=((X2-X1)**2+(Y2-Y1)**2)**0.5
C IPTS=IFIX(D/W)+1
10 SX=(X2-X1)/FLOAT(IPTS-1)
SY=(Y2-Y1)/FLOAT(IPTS-1)
DO 15 I=1,IPTS
X(I)=X1+FLOAT(I-1)*SX
15 Y(I)=Y1+FLOAT(I-1)*SY
C
C PERFORM INTERPOLATION
C
READ (5,51) IDIM,JDIM
DO 20 K=1,4
20 READ (5,52) (Z2(I,K),I=1,IDIM)
J=2
Y0=Y0+FLOAT(IDIM-1)*W
DO 25 L=1,IPTS
IX=IFIX((X(L)-X0)/W)+1
IY=IFIX((Y0-Y(L))/W)+1
XREL=((X(L)-X0)/W)-FLOAT(IFIX((X(L)-X0)/W))
YREL=((Y0-Y(L))/W)-FLOAT(IFIX((Y0-Y(L))/W))
IF (IX.LT.2) GO TO 55
IF (IX.GT.IDIM-2) GO TO 55
IF (IY.LT.2) GO TO 55
IF (IY.GT.JDIM-2) GO TO 55
30 IF (IY.EQ.J) GO TO 40

```

## Appendix G

PAGE 8882

```

DO 35 K=1,3
DO 35 I=1, IDIM
35 Z2(I,K)=Z2(I,K+1)
J=J+1
READ (5,52) (Z2(I,J), I=1, IDIM)
GO TO 38
48 DO 45 K=1,4
Z1=Z2(IX-2+K,1)
Z2=Z2(IX-2+K,2)
Z3=Z2(IX-2+K,3)
Z4=Z2(IX-2+K,4)
45 CALL AVTAN (Z1,Z2,Z3,Z4,YREL,ZT(K))
CALL AVTAN (ZT(1),ZT(2),ZT(3),ZT(4),XREL,ZT)
DO 58 K=1,4
Z1=Z2(IX-1,K)
Z2=Z2(IX,K)
Z3=Z2(IX+1,K)
Z4=Z2(IX+2,K)
58 CALL AVTAN (Z1,Z2,Z3,Z4,XREL,ZT(K))
CALL AVTAN (ZT(1),ZT(2),ZT(3),ZT(4),YREL,ZT2)
Z(L)=ZT1+ZT2)/2
IW(L)=1
GO TO 25
55 Z(L)=B
IW(L)=0
25 CONTINUE
C
C      OUTPUT DATA (DEPENDS UPON WHETHER THE DIRECTION OF THE CUT WAS REVERSED)
C
L=B
68 L=L+1
IF (IW(L) EQ 0) GO TO 68
GRAD=Z(L+1)-Z(L)
LMAX=L-1
DO 65 LL=1,LMAX
65 Z(LL)=Z(L)-FLOAT(L-LL)*GRAD
78 L=L+1
IF (IW(L) EQ 1) GO TO 78
GRAD=Z(L-1)-Z(L-2)
DO 75 LL=L,IPTS
75 Z(LL)=Z(L-1)-GRAD*FLOAT(LL-L+1)
WRITE (6,61) IPTS
IF (IDIRN EQ 1) GO TO 85
DO 88 I=1,IPTS
II=IPTS+1-I
88 WRITE (6,62) X(II),Y(II),Z(II)
STOP
85 DO 98 I=1,IPTS
98 WRITE (6,62) X(I),Y(I),Z(I)
STOP
C
C      INPUT/OUTPUT FORMATS
C
11 FORMAT (3F6 2)
12 FORMAT (4F6 2)
13 FORMAT (14)
51 FORMAT (I3,1X,I3)
52 FORMAT (12F6 1)
61 FORMAT (14)
62 FORMAT (3F6 1)

END
SUBROUTINE AVTAN (Z1,Z2,Z3,Z4,YY,Z2)
C
C      THIS SUBROUTINE IS THE AVERAGE TANGENTS ALGORITHM FOR UNITY
C      SPACED DATA POINTS
C
A1=(Z3-Z1)/2
A2=(Z4-Z2)/2
D2=Z3-Z2
A=(A1+A2-2*B*DZ)
B=(3*B*DZ-A2-2*B*A1)
C=A1
ZZ=A*YY+3*B*YY+2+C*YY+Z2
RETURN
END

```

Fig 1

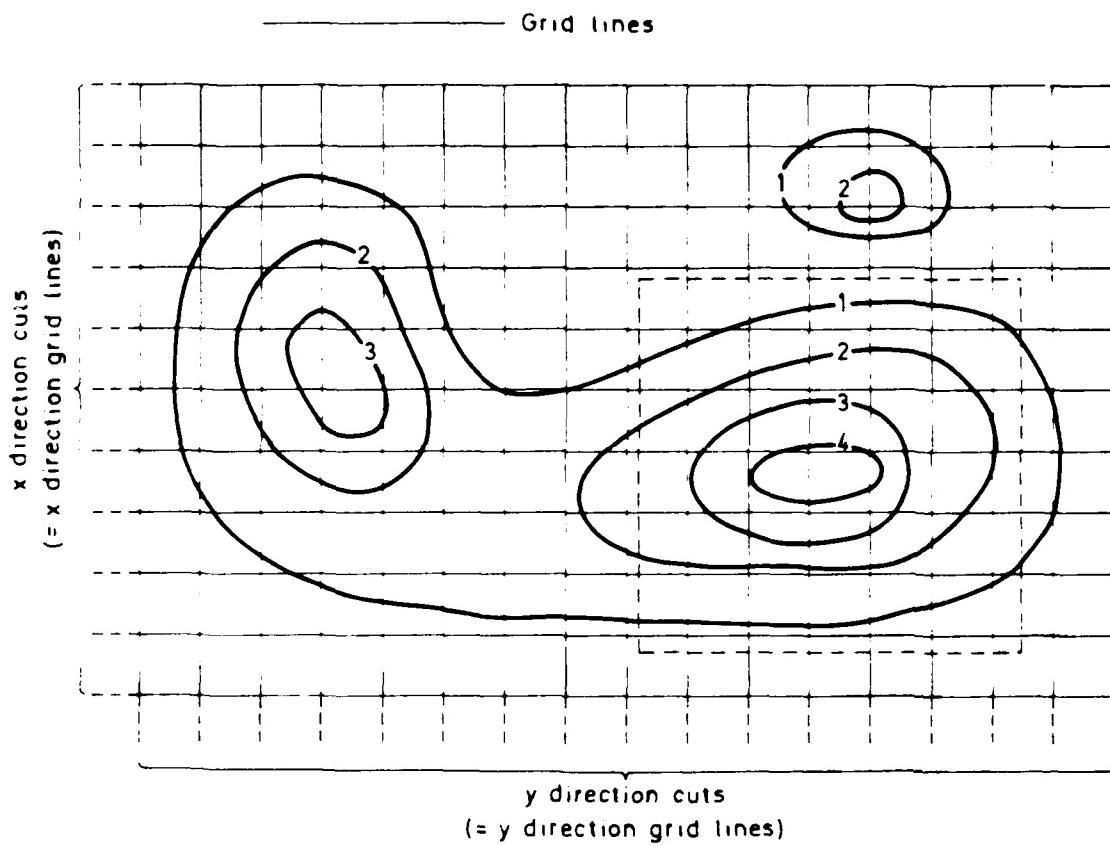


Fig 1 Geometry of program MAPGRID

Fig 2

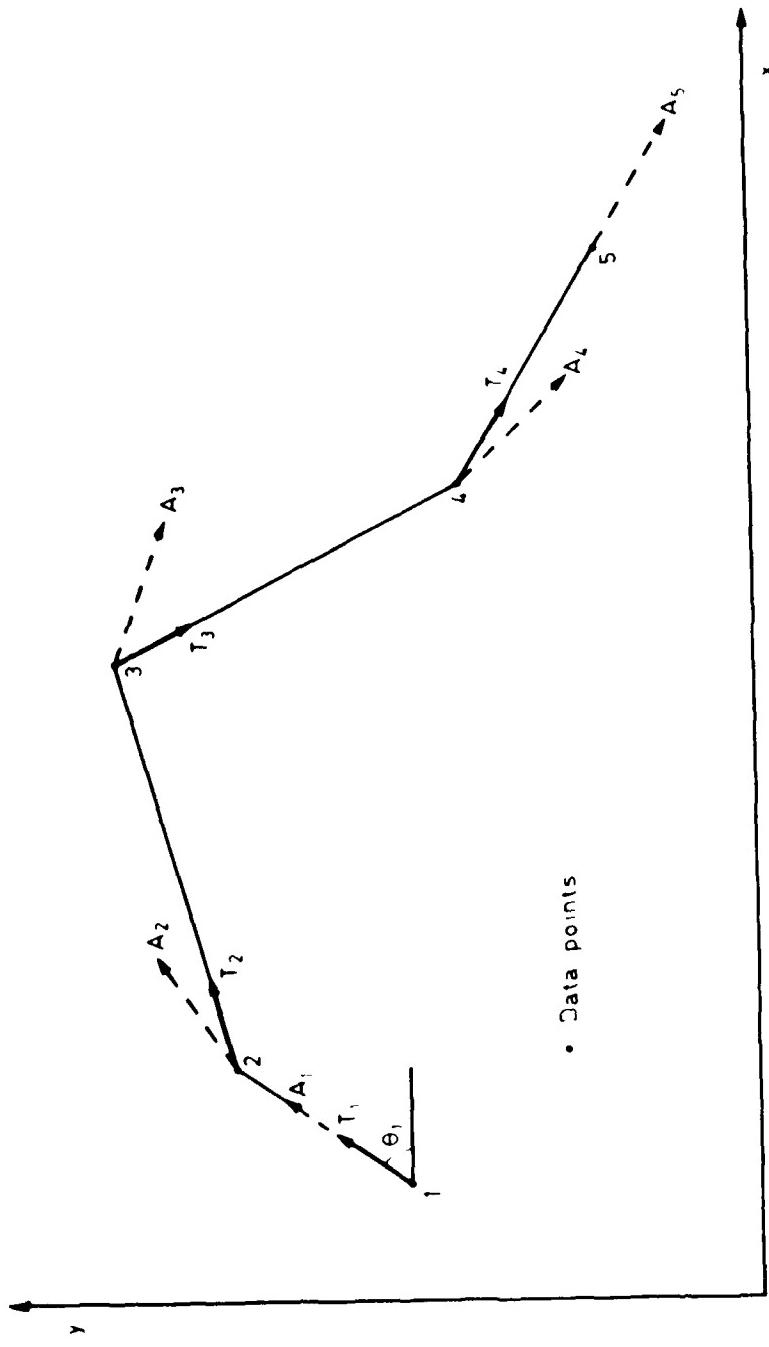


Fig 2 Geometry of the average tangents algorithm

Fig 3a&b

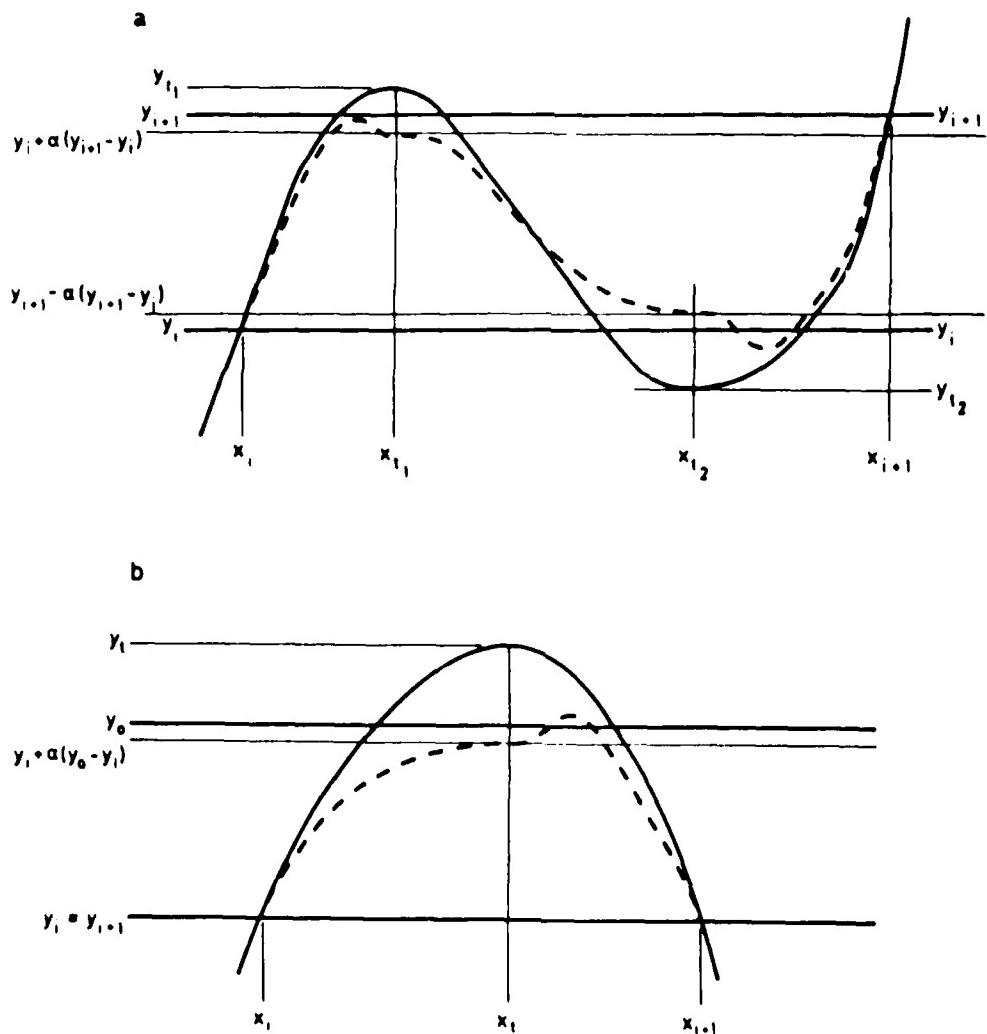


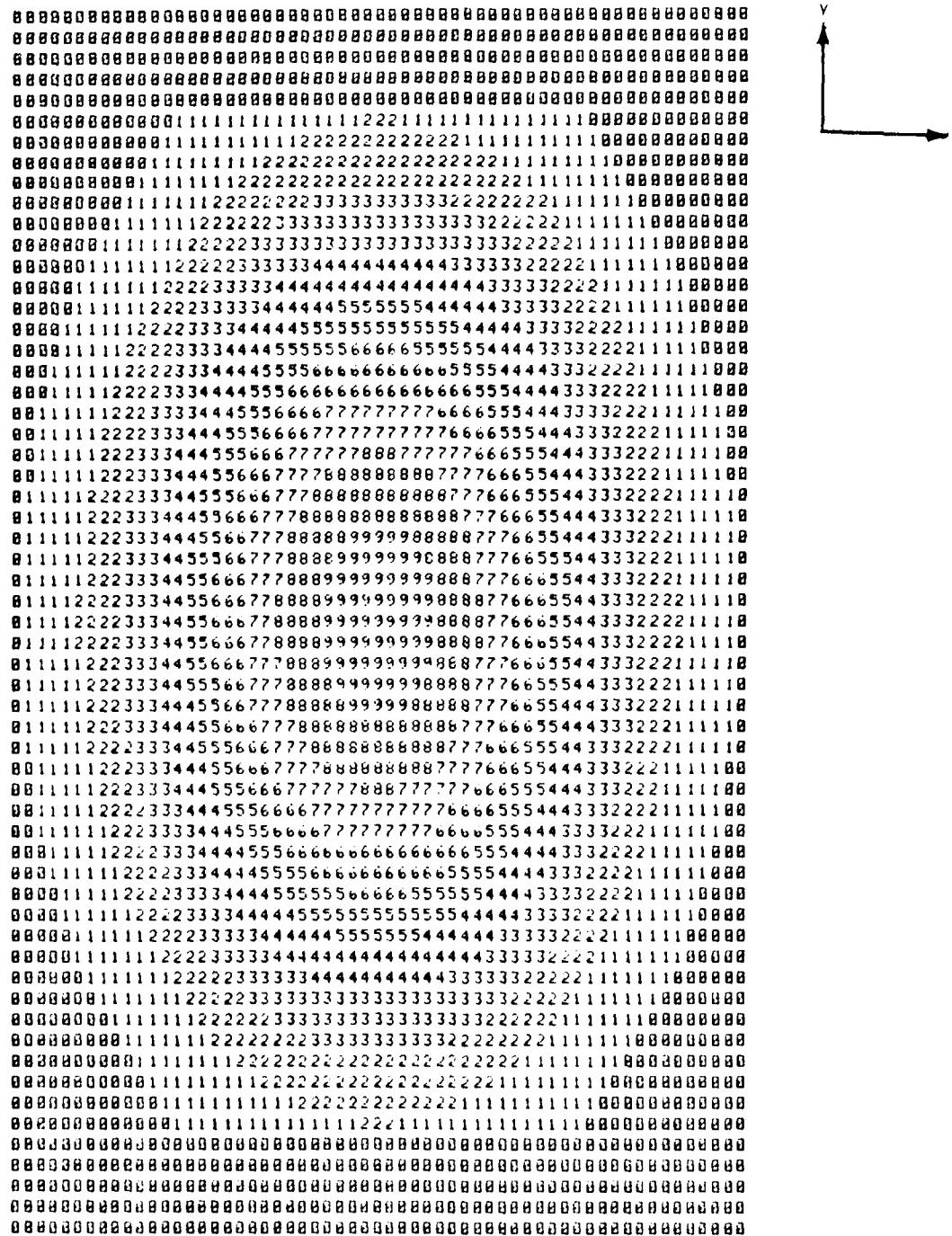
Fig 3a&b The interpolation function at turning points

Fig 4

Numbers shown represent (grid value + 25) rounded to the lower integer.

**Fig 4** Grid produced by taking cuts in y direction

Fig 5



Numbers shown represent (grid value - 25) rounded to the lower integer

Fig 5 Grid produced by taking cuts in x direction

**Fig 6**

**Fig 6** Magnitude of the differences between the program and the analytic grids using method 1 section 4.4

Fig 7

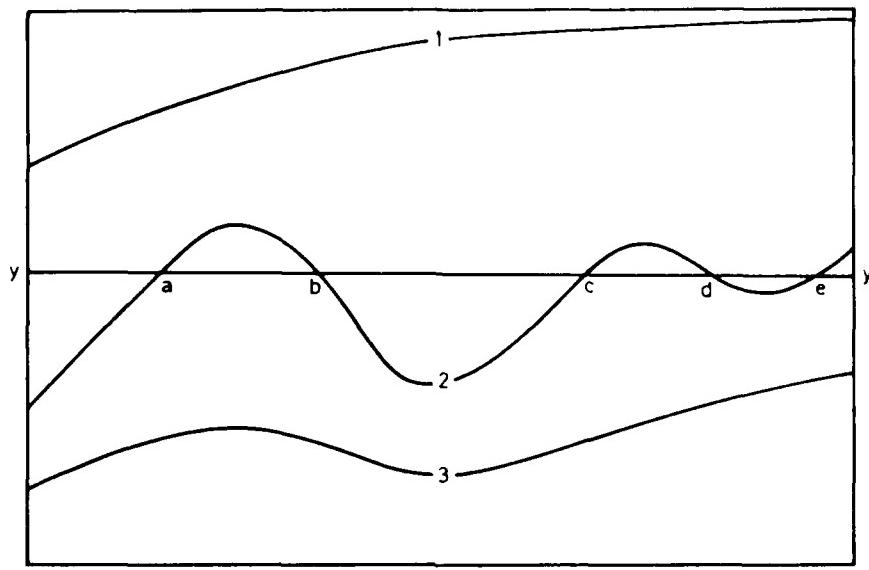
The image shows a large grid of binary digits (0s and 1s) arranged in a rectangular pattern. The grid is composed of approximately 40 columns and 40 rows of characters. The characters are black on a white background. The pattern of 0s and 1s is mostly random, with some local clustering of similar values. There are several horizontal and vertical streaks of higher density, particularly in the center and towards the bottom right. The overall appearance is that of a binary matrix or a digital representation of data.

Fig 7 Magnitude of the differences between the program and the analytic grids using method 2 section 4.4

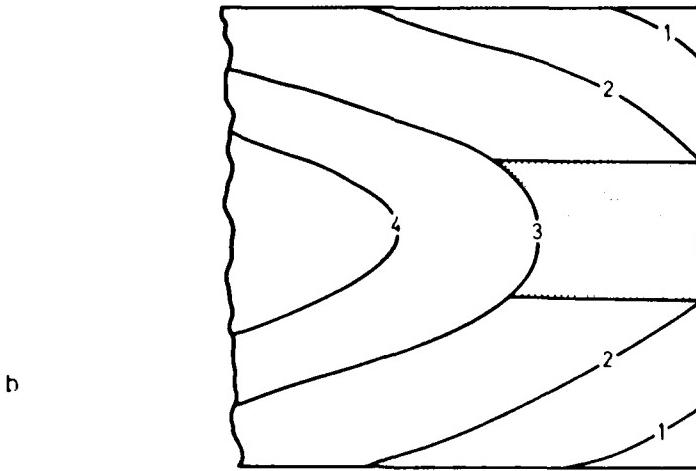
Fig 8

**Fig 8** Magnitude of the differences between the program and the analytic grids using method 3 section 4.4

Fig 9a&b



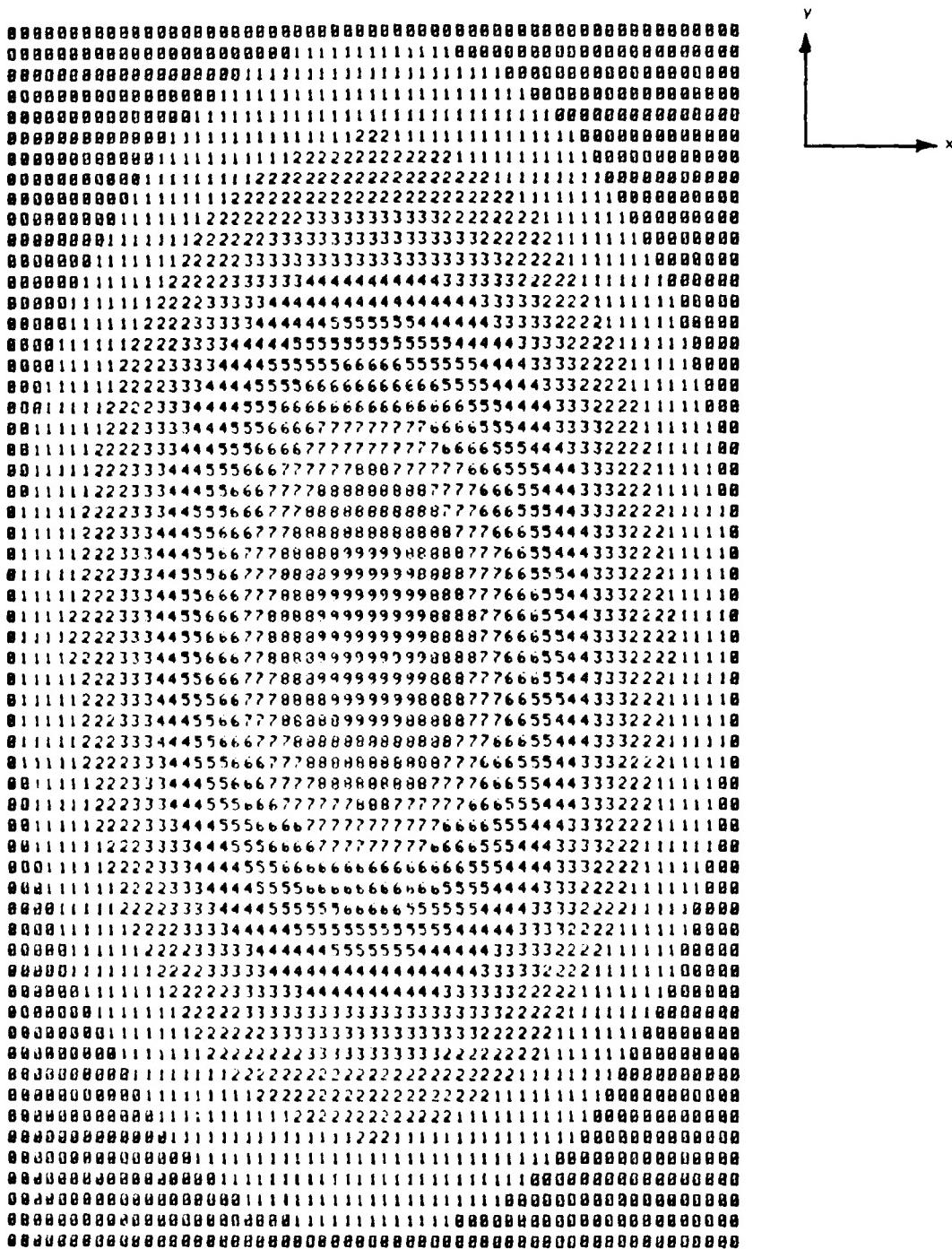
a



b

Fig 9a&b Examples of maps (see section 4.4 for explanation)

Fig 10



Numbers shown represent (grid value : 25) rounded to the lower integer

Fig 10 Final grid produced by combining those shown in Figs 4 and 5 using method 3 section 4.4

Fig 11

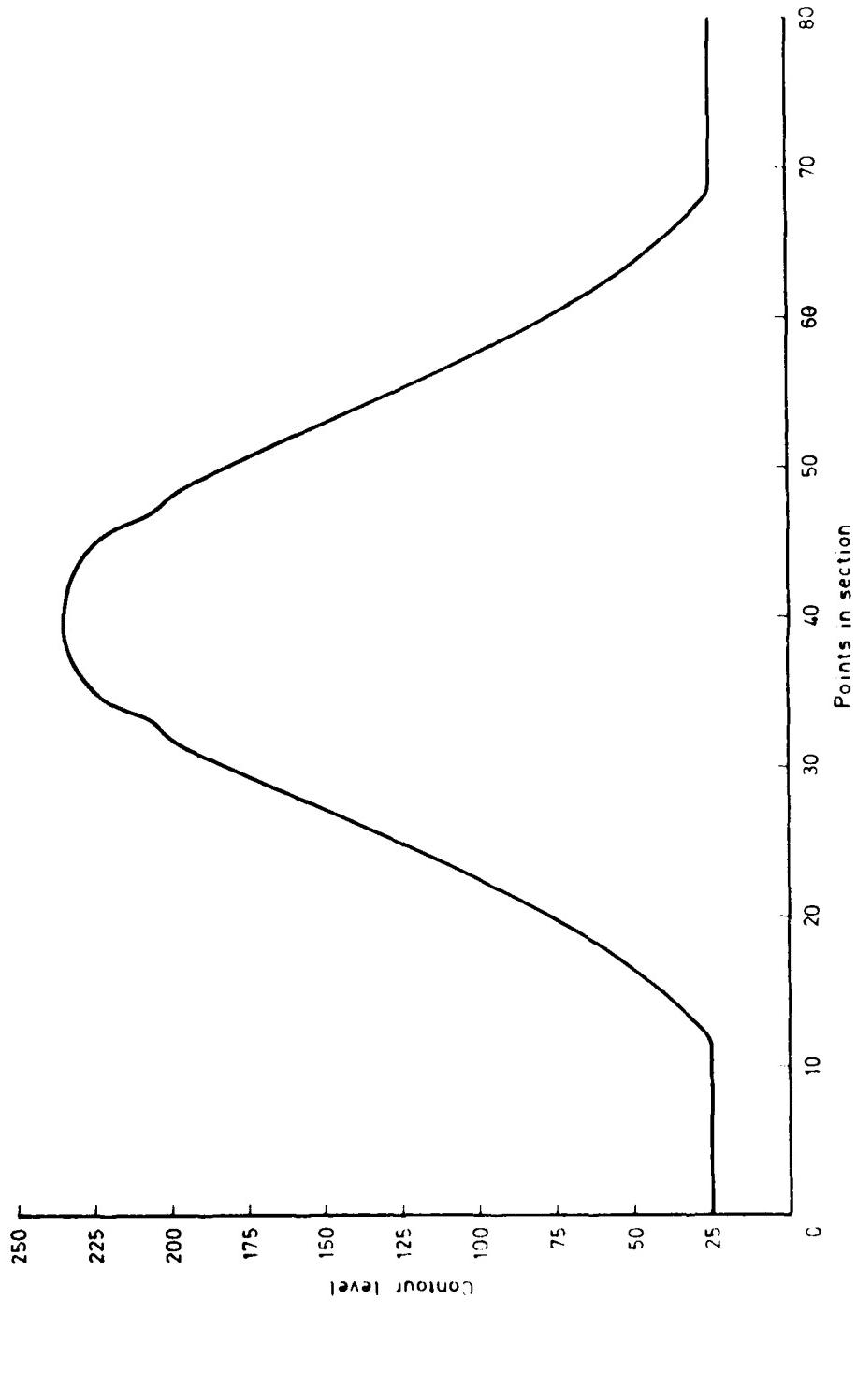


Fig 11 Diagonal cross-section through the grid shown in Fig 10

TR 80110

Fig 12

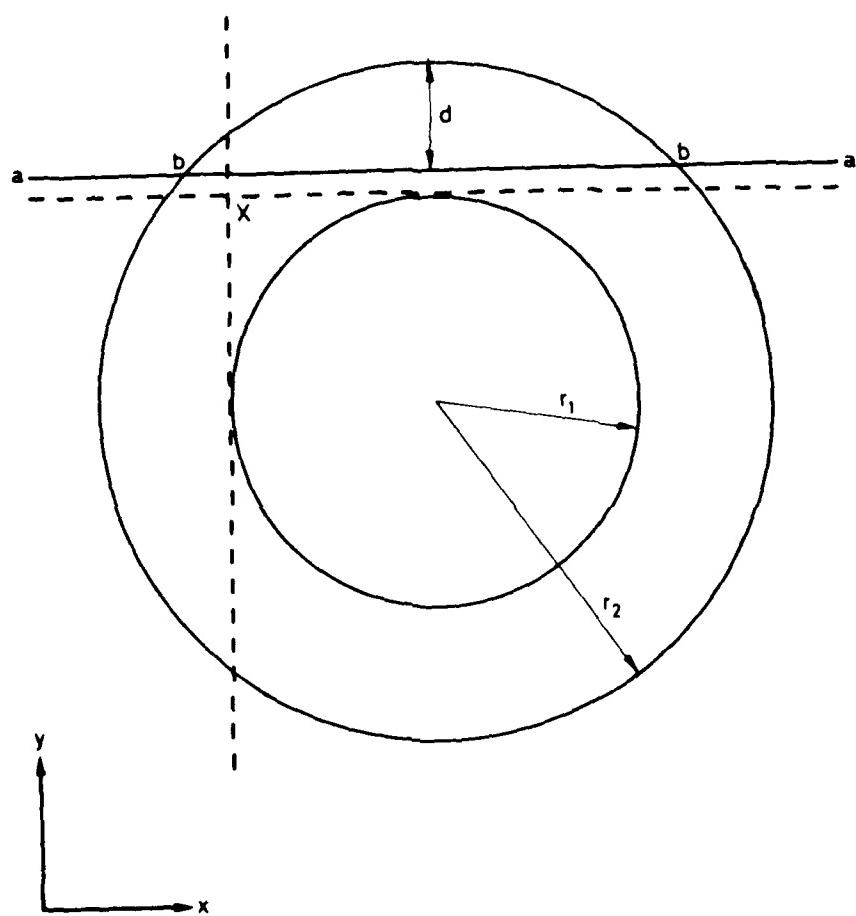


Fig 12 Geometry of operation of program MAPGRID on circular contours

**Fig 13**

Numbers shown represent (weighting value  $\times 20$ ) rounded to the lower integer

**Fig 13** Weighting values associated with each point of the final grid

Fig 14a-d

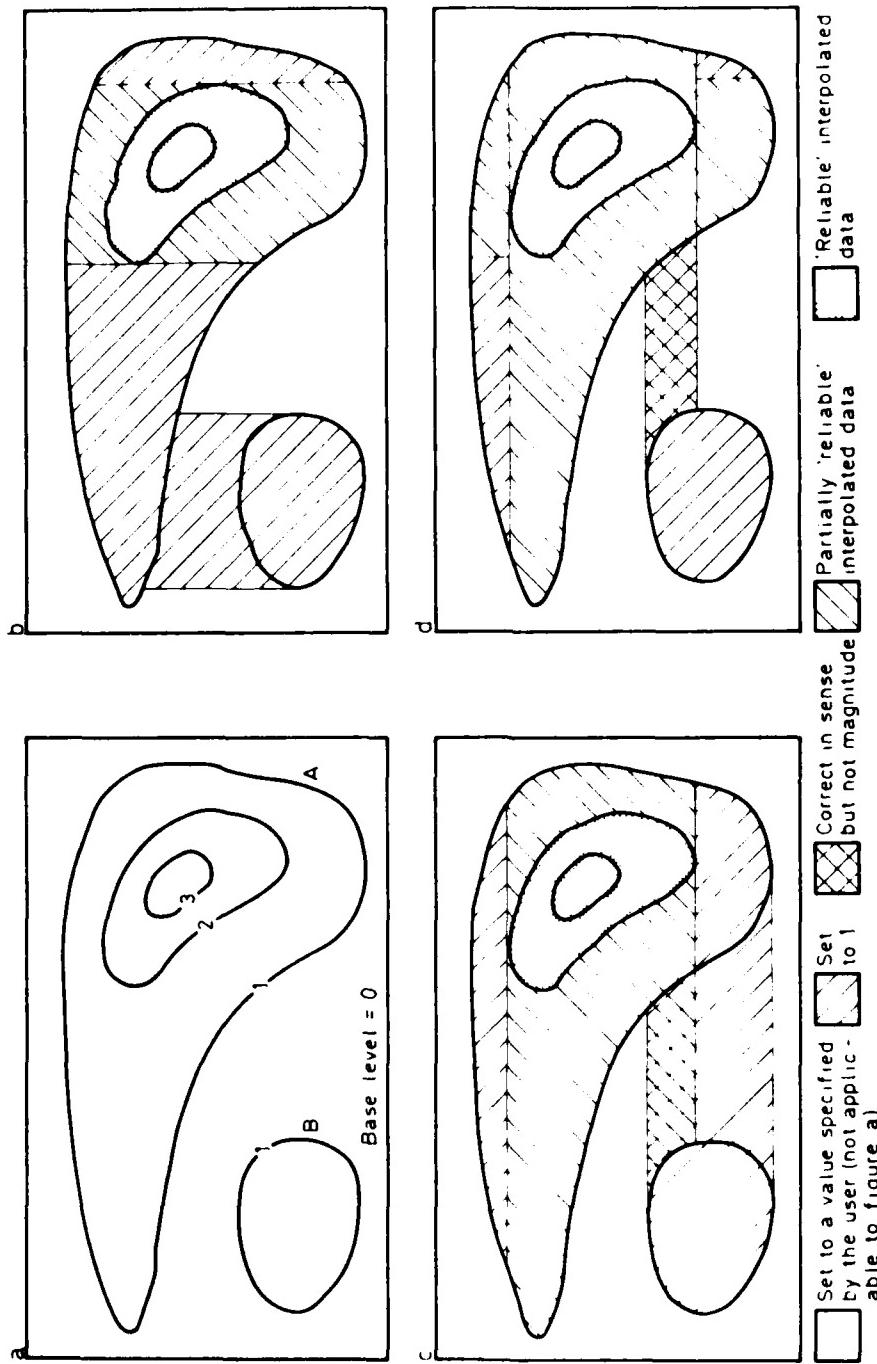
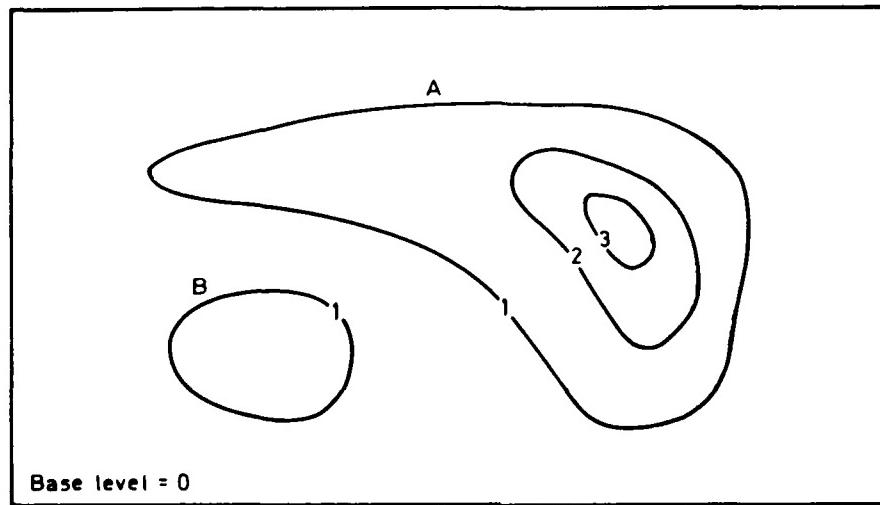
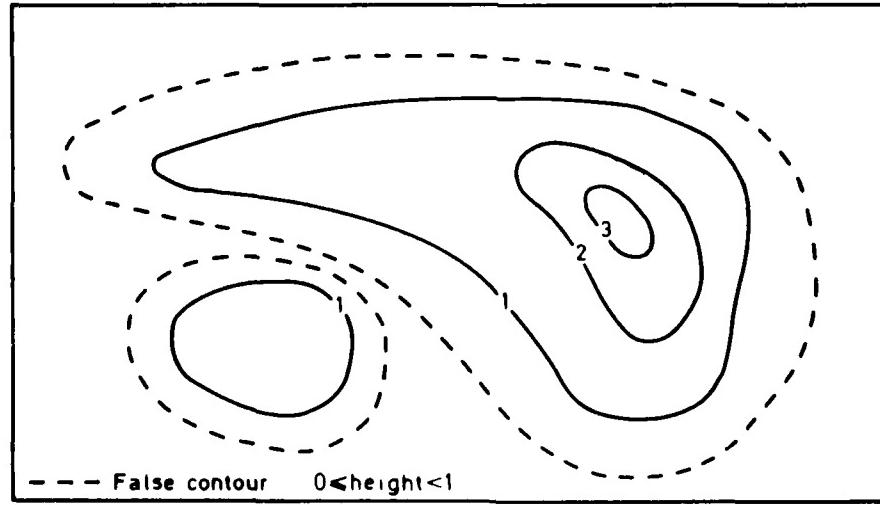


Fig 14a-d Operation of program MAPGRID on a contour map consisting of two sets of contours

Fig 15a&b



a Original contour map



b Contour map with false contours added

Fig 15a&b Introduction of false contours

Fig 16

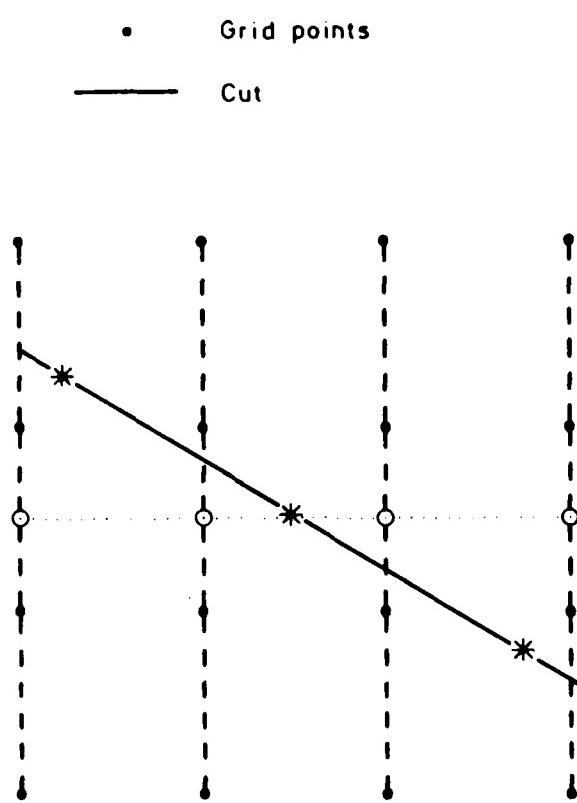


Fig 16 Geometry of the interpolation performed by program SECTION

Fig 17

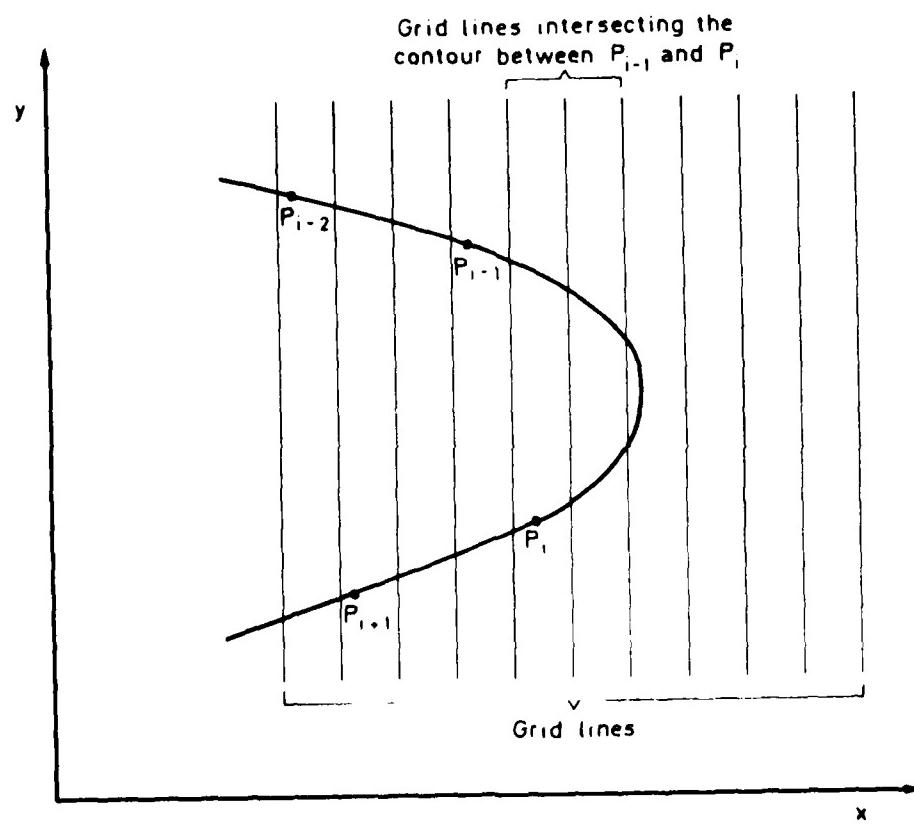


Fig 17 Grid lines crossing a contour when a contour changes direction

## REPORT DOCUMENTATION PAGE

Overall security classification of this page

**UNLIMITED**

As far as possible this page should contain only unclassified information. If it is necessary to enter classified information, the box above must be marked to indicate the classification, e.g. Restricted, Confidential or Secret.

1. DRIC Reference (to be added by DRIC)	2. Originator's Reference RAE TR 80110	3. Agency Reference N/A	4. Report Security Classification/Marking <b>UNCLASSIFIED</b>
5. DRIC Code for Originator 7673000W	6. Originator (Corporate Author) Name and Location Royal Aircraft Establishment, Farnborough, Hants, UK		
5a. Sponsoring Agency's Code N/A	6a. Sponsoring Agency (Contract Authority) Name and Location N/A		
7. Title Computer based technique for converting a contour map into an equispaced grid of points			
7a. (For Translations) Title in Foreign Language			
7b. (For Conference Papers) Title, Place and Date of Conference			
8. Author 1. Surname, Initials Roberts, P.A.	9a. Author 2	9b. Authors 3, 4 ....	10. Date September 1980   Pages 65   Refs. -
11. Contract Number N/A	12. Period N/A	13. Project	14. Other Reference Nos. Space 586
15. Distribution statement (a) Controlled by - Head of Space Department, RAE (RAL) (b) Special limitations (if any) -			
16. Descriptors (Keywords)		(Descriptors marked * are selected from TEST) Interpolation. Contour maps. Digital maps.	
17. Abstract A detailed description is given of a technique for converting a contour map into an equispaced grid of points. A full description is given of program MAPGRID which converts digitized contour data into such a grid of points.			

FSP/01

END

DATE  
FILMED

581

DTIC